David H. Ahl
COMMODORE 64
idea
BOOK
IL "LIBROIDEA"
SUL COMMODORE 64
CON 50
PROGRAMMI EDUCATIVI
PRONTI
Editrice Il Rostro

COMMODORE 64

Idea Book

David H. Ahl

COMMODORE 64
Idea Book

Il "Libroidea"
sul COMMODORE 64
con 50 programmi educativi pronti

Editrice Il Rostro

Indice

Chi è l'autore

David H. Ahl si è laureato presso la Cornell University e la Carnegie-Mellon University ed ha poi svolto lavori nel campo della psicologia educativa all'Università di Pittsburgh.
Prestò servizio presso l'Army Security Agency, lavorò poi come consulente al Management Science Associates e come ricercatore all'Educational Systems Research Institute.
All'inizio del 1970 passò alla Digital Equipment Corporation. In qualità di Direttore della linea di prodotti educativi, egli formulò il concetto di un sistema educativo computerizzato, composto di hardware, software e courseware (cioè l'insieme di materiale didattico necessario per una determinata funzione educativa), contribuendo a portare DEC ad una posizione leader nel mercato educativo.
D.H. Ahl passò poi all'AT&T nel 1974, come Direttore del marketing educativo, fu in seguito promosso Direttore marketing delle comunicazioni per il settore aziendale che sarebbe poi diventato American Bell. Contemporaneamente egli fondò la rivista *Creative Computing*, come hobby, nel 1974.
Data la crescita di Creative Computing, Mr. Ahl lasciò nel 1978 l'AT&T per dedicarsi a tempo pieno alla sua rivista, che è oggi il numero 1 nel campo del software e delle applicazioni. Nel gennaio 1980, Ahl fondò la rivista SYNC e, nel corso di questi anni, ha acquisito o fondato diverse altre pubblicazioni.
Ahl è autore o editore di 16 libri e oltre 150 articoli riguardanti l'uso del computer. È spesso chiamato come oratore o moderatore in conferenze educative e professionali.

Prefazione

Benché già Charles Babbage concepisse varie idee relative a "macchine" di calcolo, i precursori dei computer di oggi furono in gran parte sviluppati all'inizio degli anni '40 come parte dello sforzo bellico. La serie di macchine criptoanalitiche Robinson sviluppate in Inghilterra nel 1941 generarono molte famiglie di computer ancor oggi in uso. Il progetto del MIT per analisi differenziali e la simulazione di volo in tempo reale condusse al Whirlwind ed infine alla riuscitissima famiglia di computer PDP (Programmed Data Processors = Elaboratori di Dati Programmabili) della DEC (Digital Equipment Corporation). E naturalmente Eniac, costruito all'Università di Pennsylvania, che fu il punto di riferimento per Univac, IBM e molti altri produttori di successo.
Molte persone oggi guardano con divertita curiosità a queste prime macchine e le considerano come dinosauri da museo. È tuttavia interessante notare come un computer di grandi dimensioni di 25 o 30 anni fa abbia all'incirca la stessa potenza di un tipico personal computer attuale. Era generalmente meno affidabile e user-friendly (*) di un personal computer e, naturalmente, costava decine di migliaia di volte in più. Perché tutto questo?
Perché un computer degli anni '50 richiedeva che un programmatore molto abile e pieno di risorse risolvesse i problemi in base alle capacità del computer. Egli non disponeva di grandi quantità di memoria, di velocità di calcolo abbagliante, di accesso casuale al disco. In altre parole, aveva più o meno da affrontare gli stessi problemi che voi avete con il vostro personal computer.
Con questo non intendiamo dire che il vostro personal computer non sia un computer a tutti gli effetti. Può certamente essere chiamato computer quanto uno degli attuali

giganti che occupano un'intera stanza. Tuttavia, a causa della memoria relativamente piccola, non può contenere un grande archivio di dati. Nè è adatto per ampi lavori di elaborazione testi o di calcoli in grosse quantità. Anche le elaborazioni grafiche altamente dettagliate sono meglio realizzate con altre macchine.
Che cosa possiamo imparare dai pionieri informatici degli anni '50, che ci possa essere d'aiuto oggi? Forse la cosa più importante è l'insegnamento di analizzare attentamente ogni problema, suddividendolo in fasi più maneggevoli e risolvendolo una fase alla volta. È anche importante determinare ciò che può essere fatto "off line", cioè a parte, e ciò che invece va realizzato sul computer.
È questo lo scopo del presente libro. Nonostante contenga oltre 50 programmi pronti per essere eseguiti, ciò che principalmente dovreste cercare di ottenere da questo libro è un approccio alla soluzione di problemi, siano essi grandi o piccoli. Alcuni tra i problemi illustrano le capacità del computer; altri individuano i suoi limiti. È importante familiarizzarsi con i punti forti così come con quelli deboli dei vostri strumenti, così da riconoscere i tipi di impieghi per cui essi sono adatti (e quelli per cui non sono adatti).
Il libro mette a fuoco principalmente le applicazioni del computer in campo matematico ed educativo. Ci sono comunque molte altre eccellenti fonti di informazione su altre applicazioni e su come fare il miglior uso del proprio computer. L'utilizzo degli approcci descritti in questo libro vi dovrebbe mettere in grado di convertire facilmente programmi ed utilizzare esempi applicativi tratti da altri libri e riviste come *Creative Computing.*
Il libro è impostato in modo da essere letto con un computer a disposizione. Benché ci sia del testo da leggere, la parte più importante sono gli esperimenti ed i problemi da provare con il vostro computer. Il testo pone molte domande alle quali dovreste cercare di dare risposta. Non ci sono risposte a queste domande e problemi, al termine del libro. Dovreste infatti essere in grado di scoprire le risposte man mano che applicate i problemi sul vostro computer.
Potrete incorporare molte delle routine e degli approcci del libro in programmi vostri, quando userete il computer per affrontare i problemi del "mondo reale". Altri programmi vi indirizzeranno semplicemente nella giusta direzione. Alcuni programmi di questo libro costituiscono invece puro divertimento. Imparate. Sperimentate. Divertitevi!

David. H. Ahl

(*) Letteralmente "amico dell'utente", questa espressione è usata per intendere un computer facile da usare per un utente non specializzato (N.d.T.).

1
Esercitazioni e pratica

Nella vita, vi sono certe cose che vanno semplicemente imparate a memoria. Alcune cose ovvie che ricadono in questa categoria sono le tavole di addizione e di moltiplicazione, l'ortografia ed il significato delle parole, il modo in cui leggere le ore ed il sistema monetario.
Tuttavia, a seconda della professione che ognuno sceglie, vi sono molte altre cose da tenere a mente. Un medico deve conoscere la corrispondenza tra le malattie ed i relativi sintomi. Un chimico deve conoscere le leggi dei gas, le proprietà degli elementi e così via. Un pilota deve comprendere immediatamente il significato di quanto legge su dozzine di strumenti.
Per memorizzare un insieme di fatti, bisogna riesaminarli ripetutamente e provare diverse variazioni. È qui che entra in gioco il computer. Esso è in grado di presentarvi innumerevoli variazioni su problemi diversi per tutto il tempo che volete. Alcuni programmi si adegueranno automaticamente al vostro livello di competenza e vi valuteranno; altri programmi presenteranno semplicemente i problemi e lasceranno a voi la valutazione sulle vostre capacità.
Vi sono quattro programmi in questo capitolo e due in quello relativo alle scienze, i quali presentano materiale sotto forma di esercitazioni e pratica. Esaminate i metodi usati in questi programmi e poi realizzate da voi alcuni programmi di esercitazione su argomenti con i quali avete problemi, oppure realizzate programmi per altri membri della vostra famiglia.

Pratica sull'addizione

Questo programma mostra una routine di addizione semplificata, a scopo di esercizio e pratica. Questo tipo di esercitazione è talvolta chiamato "Istruzione assistita dall'elaboratore" (CAI - Computer Aided Instruction), benché il CAI possa essere applicabile ai tutotorial così come ad altri approcci del genere.
Quando viene lanciato, il programma per prima cosa chiede "Numero di cifre?" Voi scrivete un numero ed ogni addendo conterrà così un'uguale o inferiore quantità di cifre.
Il programma vi presenterà qualunque problema pratico che gli specificherete. Esso presenterà ogni problema in successione e non passerà al problema successivo finché quello attuale non abbia ricevuto una risposta corretta. Quando anche l'ultimo problema ha ricevuto una risposta corretta, viene stampato il punteggio insieme ad un commento appropriato.
In un programma come questo sono possibili molti miglioramenti ed estensioni. Ad esempio, potreste modificarlo in modo che venga fornita all'utente la risposta corretta quando ad un problema siano già state date due (o tre) risposte errate.
Una modifica più complessa potrebbe essere di intervenire sul programma in modo che presenti differenti tipi di problemi aritmetici, come sottrazioni, moltiplicazioni e divisioni.
Alcune di queste modifiche sono state fatte nel prossimo programma.

```
5 printchr$(14):rem set minuscole/maiuscole
10 printchr$(147);"Pratica sull' addizione"
20 input"n' cifre ";n
30 input"n' problemi ";p
40 w=0
50 x=x+1
60 q=0
70 a=int(10↑n*rnd(1))+1
80 b=int(10↑n*rnd(1))+1
90 print
100 print tab(8-int(log(a)/log(10)+1));a
110 print tab(7-int(log(b)/log(10)+1));"+";b
120 print" --------"
130 c=a+b
140 printtab(7-int(log(c)/log(10)+1));
150 inputg
160 ifg=cthen230
170 q=q+1
180 ifq>1then200
190 w=w+1
200 print:print"Errato !  Riprova..."
220 goto90
230 ifx>=pthen280
250 print"Eccone un' altra..."
270 goto50
280 print:print:print"Hai dato "p-w" risposte esatte"
290 ifw<.22*pthen320
300 print"e "w" errate."
310 end
320 print:print"Ottimo lavoro !":end
```

```
Pratica sull' addizione
n' cifre ? 3
n' problemi ? 2

      189
    + 973
 --------
     1162
Eccone un' altra...

      777
    + 418
 --------
     1195


Hai dato  2  risposte esatte

Ottimo lavoro !
```

Pratica sull'addizione, adeguata al livello di abilità

Uno dei maggiori svantaggi di molti programmi di esercitazione e pratica è che essi tendono ad essere noiosi, oppure frustranti, a seconda dell'abilità dell'utente, in rapporto al livello del materiale usato. Per evitare questo inconveniente è necessario un metodo che adegui la difficoltà dei problemi all'abilità dell'utente.
Idealmente, un sistema del genere dovrebbe tenere in maggiore considerazione il risultato più recente, senza però ignorare i risultati precedenti. Dovrebbe consentire all'utente di avanzare verso problemi più difficili rispetto al livello che egli può padroneggiare. Dovrebbe inoltre continuare a fornire della pratica sui problemi già padroneggiati.
Alcuni prodotti software in commercio raggiungono questi obiettivi secondo schemi tradizionali, ad esempio determinando in quali tipi di problemi uno studente dovrebbe esercitarsi, usando un complicato sistema di registrazione ed adeguamento computerizzato del punteggio.
L'approccio usato qui è più innovativo; usa infatti una singola misura per ogni tipo di problema — chiamata "livello stimato di abilità" — che raggiunge tutti gli obiettivi prima indicati.
Come funziona? L'ultimo problema presentato conta il 10% del punteggio totale se è stato risolto correttamente ed era al di sopra dell'attuale livello di abilità dell'utente, oppure aveva ricevuto una risposta errata ed era al di sotto dell'attuale livello di abilità dell'utente. In ogni altro caso il punteggio non viene modificato. Tutto ciò può essere più facilmente visualizzato secondo il seguente schema:

	Risposta Corretta	*Risposta Errata*
Problema superiore al livello di abilità	Aumenta il livello di abilità	Ignorato
Problema inferiore al livello di abilità	Ignorato	Diminuisce il livello di abilità

A prima vista può apparire complesso ed un po' goffo, tuttavia esso significa in realtà che lo studente è premiato se risolve un problema superiore al suo livello attuale, ma non è penalizzato se non ci riesce. D'altra parte egli è penalizzato se non riesce a risolvere un problema inferiore al suo livello attuale, senza essere però premiato se ci riesce.
Ogni problema influisce un po' sul livello di abilità stimato, ed i problemi più recenti vengono valorizzati più pesantemente. Se il livello attuale di uno studente è L ed il

livello del più recente problema da inserire nella media è P, allora la formula di media è semplicemente:

$$L = .9L + .1P$$

Prima di poter scrivere il programma dobbiamo ancora assegnare un livello di abilità ad ogni problema presentato. Sfortunatamente, ciò può variare in rapporto al sistema scolastico locale, al libro di testo usato ed al metodo di insegnamento. Inoltre in un piccolo computer non può essere memorizzato un grosso insieme di dati, perciò è bene individuare un metodo semplice per determinare il livello di abilità per i diversi problemi. Un approccio lineare consiste nel presentare problemi che siano compresi entro mezzo punto in più o in meno rispetto al livello a cui si trova lo studente. Per uno studente al livello 3.2 l'ambito di problemi da presentare sarebbe allora da 2.7 a 3.7.

Come generare i problemi giusti? Consideriamo un tipo di abilità: l'addizione in verticale. È normalmente introdotta al primo livello, e continua attraverso il livello 4 (in effetti fino a 4.9). Il problema più semplice in questo programma è 1 + 1 mentre il più difficile è 999 + 999. Poichè l'apprendimento non è un processo lineare (inizialmente è lento e poi progredisce rapidamente), si può usare una formula esponenziale. Ad esempio:

$$\text{Addendo} = 1.73 \text{ x } (\text{Livello di abilità})^4$$

oppure:

$$\text{Livello di abilità} = \sqrt[4]{\text{Addendo} / 1.73}$$

Ciò significa che gli studenti ai vari livelli lavoreranno con i seguenti addendi massimi:

Livello di abilità	*Addendo*
1.0	11
2.0	27
3.0	140
4.0	442
4.9	997

A questo punto è relativamente semplice, benché un po' tedioso, collegare insieme tutti questi elementi in un programma per computer.

Alcune note sul programma. La variabile G2 è il livello del programma, che è sempre entro mezzo punto di differenza da G1, il livello attuale dello studente. La complicata linea 290 genera un livello medio di abilità variabile.

La registrazione del livello di abilità ed il suo trasferimento alla prossima lezione è un processo manuale. Su un computer con una memoria di massa permanente, tutto ciò potrebbe essere conservato nel sistema.

Ci sono molte possibili modifiche ed estensioni del programma. Per esempio si potrebbero presentare tipi diversi di problemi, come addizioni orizzontali, sottrazioni verticali e orizzontali, così come moltiplicazioni, divisioni e problemi di frazioni.

```
5 printchr$(14):rem set minuscolo/maiuscolo
10 w=0
20 printchr$(147);"Ciao. Per terminare ricordati di rispon-dere con 9999."
30 input"Livello ";g1
40 g2=g1-.5*rnd(0)
50 r=int(2+1.73*g2↑4)+1
60 a=int(100*g2*rnd(1))+1
70 ifa>rthen60
80 b=r-a
90 print:printtab(8-int(log(b+1)/log(10)+1));b
100 printtab(7-int(log(a+1)/log(10)+1));"+";a
110 print" --------"
120 r=a+b
130 printtab(7-int(log(r)/log(10)+1));
140 inputg
150 ifg=9999then320
160 ifg=rthen260
170 w=w+1
180 ifw>1then210
190 print:print"Errato ! Riprova..."
200 goto90
210 print:print"Hai sbagliato due volte !                    La risposta e' ";r
230 w=0
240 ifg2>g1then300
250 goto290
260 w=0
270 print:print"Corretto!";
280 ifg2<g1then300
290 g1=.9*g1+.1*sqr(sqr(r/2/1/73))
300 print:print"Eccone un' altra...":print
310 goto40
320 printchr$(147)
330 print"Okay..."
340 g1=(int(100*g1))/100
350 print"Nuovo livello : ";g1
```

```
Ciao. Per terminare ricordati di rispondere con 9999.
Livello ? 3

       46
     + 64
 --------
      110

Corretto!
Eccone un' altra...


       36
     + 34
 --------
       70

Corretto!
Eccone un' altra...


       62
     + 37
 --------
       99

Corretto!
Eccone un' altra...


        0
    + 124
 --------
      124

Corretto!
Eccone un' altra...


       52
     + 88
 --------
      140

Corretto!
```

Problemi di tempo/velocità/distanza

Il computer può presentare non solo semplici problemi di esercizio e pratica numerici, ma anche problemi esprimibili a parole.
In questo programma la formula che mette in relazione tempo, velocità e distanza è stata applicata ad un problema riguardante un'automobile ed un treno.
Il problema può essere espresso come segue. Un'auto che viaggia ad una velocità di C miglia all'ora (il computer genera un valore intero compreso tra 40 e 65) può percorrere una certa distanza in D ore (il computer genera un valore intero tra 5 e 20) in meno rispetto ad un treno che viaggia a T mph (il computer genera un valore intero tra 20 e 39). Quanto tempo impiega l'auto a percorrere l'intero tragitto? Risolvendo le due equazioni simultanee si ottiene un'unica equazione risolutrice mostrata nella linea 60.
Osservate l'espressione contenuta nella linea 70. Essa calcola la differenza percentuale tra la risposta esatta e quella data dall'utente.
Osservate anche che il computer calcola la risposta esatta nella riga 60 e la scrive sullo schermo alla linea 130. Questa risposta può avere molte cifre decimali; il programma è scritto in modo da arrotondarla a due cifre decimali. Se non fosse stato aggiunto 0.5 nella linea 60, il numero sarebbe stato troncato e non arrotondato. Questo è uno schema di calcolo importante che troverete in molti altri programmi nel corso di questo libro.
Provate a considerare altri problemi che possono essere usati come base per questo tipo di esercitazione pratica. Gli insegnanti, ad esempio, potrebbero richiedere agli studenti di scrivere da sè alcuni programmi di esercitazione e pratica. Problemi diversi potrebbero essere dati ad uno o più studenti come base per la stesura di un programma.
Quando i programmi sono stati scritti, gli studenti possono provare i programmi realizzati dagli altri compagni di classe. Questo approccio assicura che ogni studente comprenda non solo il tipo di problema che gli è stato assegnato, ma anche altri tipi di problemi. Questa è una tecnica efficace per stimolare l'interesse e per imparare a risolvere quesiti espressi in forma verbale.

```
5 printchr$(14):rem set minuscolo/maiuscolo
10 c=int(rnd(1)*25)+40
20 d=int(rnd(1)*15)+5
30 t=int(rnd(1)*19)+20
35 printchr$(147);
40 print"Un auto a"c" mph fa un viaggio in"d
45 print"ore in meno di un treno a "t" mph"
50 print:input"Ore d'auto ";a
60 v=int(.5+100*(d*t/(c-t)))/100
70 e=int(abs((v-a)*100/a)+.5)
80 print
90 ife>5then120
100 print"Bene. Sei stato sotto al "e"% ."
110 goto130
120 print"Spiacente. Sei fuori del "e"% ."
130 print"Tempo esatto = "v
140 print:input"Ancora (s/n) ";z$
150 ifz$="s"orz$="S"then10
160 print:print"Okay... ciao!"
170 end
```

```
Un auto a 55  mph fa un viaggio in 17
ore in meno di un treno a  25  mph

Ore d'auto ?  2

Spiacente. Sei fuori del  136 % .
Tempo esatto =  14.17

Ancora (s/n) ? s
Un auto a 51  mph fa un viaggio in 19
ore in meno di un treno a  31  mph

Ore d'auto ?  4

Spiacente. Sei fuori del  391 % .
Tempo esatto =  29.45

Ancora (s/n) ? s
Un auto a 46  mph fa un viaggio in 10
ore in meno di un treno a  20  mph

Ore d'auto ?  4

Spiacente. Sei fuori del  28 % .
Tempo esatto =  7.69

Ancora (s/n) ? s

Un auto a 54  mph fa un viaggio in 8
ore in meno di un treno a  29  mph

Ore d'auto ?  3

Spiacente. Sei fuori del  55 % .
Tempo esatto =  9.28

Ancora (s/n) ? n

Okay... ciao!
```

Problemi di cinematica

La cinematica studia la dinamica dei corpi in movimento, indipendentemente da considerazioni di massa e forza. Come molti altri tipi di problemi, anche questi possono essere generati e presentati dal computer per impratichirsi nella loro soluzione.

Questo programma presenta un semplice problema cinematico da risolvere. Il computer genera un nuovo valore per ogni problema. Si tratta di questo. Una palla (o qualunque altro oggetto) viene lanciata verso l'alto ad una velocità di V metri al secondo (il computer genera un valore intero compreso tra 5 e 40). L'utilizzatore deve allora calcolare tre fattori riguardanti il risultante volo della palla: altezza massima raggiunta, tempo impiegato per ritornare al suolo e velocità dopo T secondi (il computer genera un tempo inferiore al tempo totale di volo).

Il beneficio fondamentale derivante dall'usare un computer per presentare problemi di questo tipo è la motivazione. I calcoli richiesti all'utilizzatore non sono diversi da quelli esposti al termine di un capitolo di un testo scolastico o da quelli assegnati come compiti a casa. Tuttavia, rispondere a questi problemi, quando essi sono presentati dal computer, sembra trasformare il lavoro in una sfida e, francamente, lo rende più divertente.

Il programma verifica ogni vostra risposta, per vedere se non si discosta oltre il 15% della risposta corretta. Se è così, la vostra risposta è considerata corretta. Potreste modificare questa percentuale così da richiedere calcoli più o meno accurati. Potreste anche modificare i calcoli effettuati dal computer per arrotondarli ad uno o due decimali.

```
5 print chr$(14):rem setta minuscole
10 q=0
20 v=6+int(35*rnd(1))
30 print chr$(147);"La palla sta salendo a";v;"metri al secondo"
40 a=.05*v↑2
50 input "Qual'e' la massima altezza";g
60 gosub 210
70 a=v/5
80 print:input "Quando scendera' (sec)";g
90 gosub 210
100 t=2+int(2*v*rnd(1))/10
110 if t>v/5 then 100
120 a=v-10*t
130 print:print "Qual'e' la velocita' dopo";t;"secondi";
140 input g
150 gosub 210
160 print:print q;"risposte esatte su 3"
170 if q>1 then print "non c'e' male."
180 print:input "Ancora (s,n)";z$
190 if z$="s" or z$="S" then 10
200 end
210 if abs((g-a)/a)<.15 then 240
220 print "Siamo ancora lontani..."
230 goto 260
240 print "Molto bene."
250 q=q+1
260 print "La risposta esatta e':";a
270 return
```

```
La palla sta salendo a 34 metri al secondo
Qual'e' la massima altezza ?  25
Siamo ancora lontani...
La risposta esatta e': 57.8000001

Quando scendera' (sec) ?  0
Siamo ancora lontani...
La risposta esatta e': 6.8

Qual'e' la velocita' dopo 5.2 secondi ? 1
Siamo ancora lontani...
La risposta esatta e':-18

 0 risposte esatte su 3

Ancora (s,n) ? s


La palla sta salendo a 12 metri al secondo
Qual'e' la massima altezza ?  16
Siamo ancora lontani...
La risposta esatta e': 7.2

Quando scendera' (sec) ?  7
Siamo ancora lontani...
La risposta esatta e': 2.4

Qual'e' la velocita' dopo 2.1 secondi ? 4
Siamo ancora lontani...
La risposta esatta e':-9

 0 risposte esatte su 3

Ancora (s,n) ? n
```

2
Risoluzione di problemi

In molti corsi scolastici i libri di testo presentano una gran varietà di strumenti per risolvere i problemi appositamente raggruppati al termine di ogni capitolo. Tipicamente questi strumenti consistono in formule, equazioni, regole e teoremi. Dopo un attento studio di questi strumenti, gli insegnanti effettuano esami che verificano la vostra abilità nel ricordarli.

Ma che si può fare quando si deve affrontare la più realistica situazione in cui non ci venga detto quale strumento sia idoneo per affrontare certi problemi o, peggio ancora, si è dimenticato del tutto come si usa una certa tecnica? Ogni speranza è persa? Certamente no, benché sembri che alcune persone lo credano.

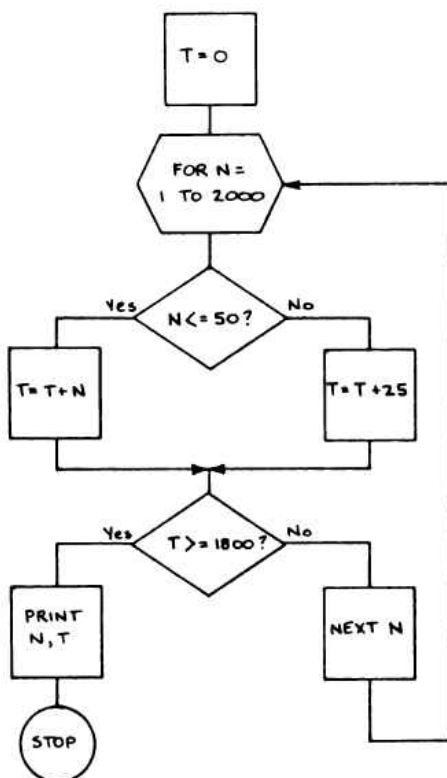
In questo capitolo, sono presentati diversi tra questi ostili strumenti. Per esempio, vi sono programmi in grado di risolvere equazioni quadratiche o esponenziali ed altri che calcolano le radici e tracciano la curva di ogni funzione.

Comunque, dovete ricordare che questi strumenti sono utili per risolvere certi problemi, ma la cosa veramente importante è comprendere la logica ed il tipo di approccio che ne sta alla base. Così, quando arriverà il momento di risolvere i problemi del mondo reale, sarete meglio preparati a fronteggiarli. Tra l'altro, molti tra i metodi e gli strumenti presentati in questo capitolo saranno usati in capitoli successivi per risolvere altri tipi di problemi.

Quanti biglietti?

Ed ecco un problema. Ad una lotteria scolastica, organizzata per raccogliere fondi, gli organizzatori hanno messo in palio un gioco elettronico, per il quale hanno speso 18.00 $. Per rendere più interessante la lotteria, gli organizzatori hanno deciso di vendere i biglietti ad un importo (in cents) pari al numero del biglietto stesso, per i biglietti numerati da 1 a 50. Per i successivi, il prezzo sarà di 25 cents ciascuno. Gli organizzatori vogliono allora sapere quanti biglietti devono vendere per pareggiare esattamente la spesa effettuata.

È probabilmente più facile visualizzare questo tipo di problemi con un diagramma di flusso (flow-chart), in cui T contiene il totale del denaro raccolto e viene incrementato man mano che i biglietti sono venduti. Il numero del biglietto è N. Quando T diviene maggiore o uguale a 18.00 $, N sarà allora la risposta cercata.

Osservate che il flow-chart contiene due elementi di decisione logica (istruzioni IF nel programma). Il primo confronta il numero del biglietto con 50; se è inferiore, il numero del biglietto è aggiunto al totale mentre, se è maggiore di 50, il totale è aumentato di 25 cents.

Il secondo elemento decisionale confronta invece l'importo complessivamente raccolto, T, con 18.00 $ (cioè 1800 cents). Se T è maggiore o uguale a 1800, ciò significa che è stata raggiunta la situazione cercata (break even point) ed i valori di N e T sono allora scritti sullo schermo.
Questo tipo di problema può anche essere risolto a mano, ma risulta piuttosto noioso a causa della ripetitività delle addizioni. Inoltre, a mano si ottiene frequentemente una risposta di 72, invece della corretta risposta, che è 71. Provate voi stessi per vedere che cosa ottenete.

```
5 print chr$(14)
10 t=0
20 for n=1 to 2000
30 if n<=50 then 60
40 t=t+25
50 goto 70
60 t=t+n
70 if t>1800 then 90
80 next n
90 print n;"biglietti venduti"
100 print " raccolti $";t/100
```

```
72 biglietti venduti
raccolti $ 18.25
```

Molti problemi possono essere correttamente e rapidamente risolti usando un computer, mediante l'analisi logica ed un flow-chart. Problemi più complessi possono richiedere di essere separati in passi successivi e richiedono flow-chart di maggiori dimensioni, ma l'approccio resta fondamentalmente lo stesso.

Eccovi ora altri due problemi da risolvere.
Il diametro di un disco long-playing è di 12 pollici. La parte centrale non utilizzata ha un diametro di 4 pollici, mentre il bordo esterno alla registrazione è di 1/2 pollice. Se vi sono 91 solchi per pollice, che distanza copre la puntina durante l'ascolto del disco?
Un cinema fa pagare 2.50 $ per l'ingresso di un adulto e 1 $ per un bambino. Alla chiusura, il cassiere conta 385 matrici di biglietti staccati e 626.50 $ in cassa. Quanti bambini sono entrati?

Il rapporto tra il bere e l'alta pressione sanguigna

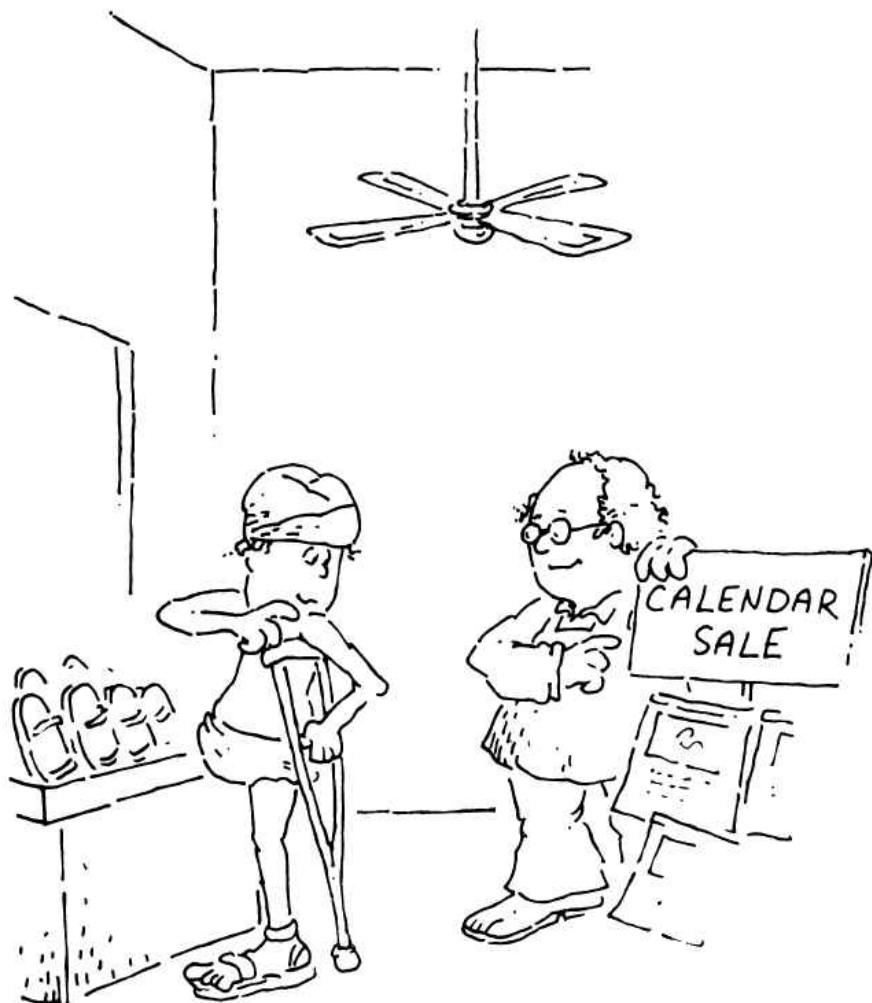
Questo programma mostra come più equazioni semplici possano essere inserite in un programma per computer, per risolvere un più difficile problema complessivo.
Ecco il problema. In un'indagine effettuata su 1000 adulti, si è rivelato che 35 di loro hanno la pressione alta. Tra questi, l'80% beve 15 o più once (oz.) di alcool alla settimana. Tra quelli che invece non soffrono di alta pressione, il 60% beve una quantità simile. Qual è la percentuale di ipertesi rispettivamente tra i bevitori e i non bevitori?
Questo problema richiede la soluzione di varie equazioni semplici. Le si può combinare in un'unica equazione complessa, ma può essere più semplice comprendere l'approccio seguito (e successivamente modificare le variabili) scrivendo un programma con cinque equazioni separate.
Detto H il numero di persone che soffrono di alta pressione, allora H = 35 (linea 10).
Detto H1 il numero di bevitori con la pressione alta si ottiene che H1 = .8 x H (linea 20).
Se L1 è il numero di bevitori con bassa pressione, allora L1 = .6 x (1000 - H). Così, il numero totale di bevitori è calcolato come D = H1 + L1. Infine, la percentuale di bevitori con alta pressione è X = H1 x 100/D.
Con questo programma il problema è risolto in un batter d'occhio. La soluzione di questo tipo di problema può essere facilmente scritta direttamente in Basic, senza alcun bisogno di un Flow-chart nè di un'analisi dettagliata. Essere capaci di riconoscere prontamente questo tipo di problema può far risparmiare una gran quantità di tempo.

```
5 print chr$(14)
10 h=35
20 h1=.8*h
30 l1=.6*(1000-h)
40 d=h1+l1
50 x=h*100/d
60 x1=h*100/1000
70 print "ALTA PRESSIONE SANGUIGNA"
80 print "POPOLAZIONE";x1;"%"
90 print "BEVITORI    ";x;"%"
```

```
ALTA PRESSIONE SANGUIGNA
POPOLAZIONE 3.5 %
BEVITORI    5.76606261 %
```

Ecco un problema che non richiede una singola equazione, bensì mette in pratica gli approcci presentati fin qui in questo capitolo. Siete in grado di risolverlo con tre sole istruzioni Basic?

All'inizio di Gennaio, un negoziante ribassa il prezzo di alcuni calendari da $ 2.00 ad un prezzo inferiore. Vende l'intero stock in un solo giorno per $ 603.77. Quanti calendari aveva?

Ed ecco un altro facile dilemma. Una città in India ha una popolazione di 20.000 persone. Il 5% di loro ha una sola gamba e la metà dei rimanenti va a piedi scalzi. Quanti sandali vengono portati nella città?

Due equazioni simultanee

Finora sono stati considerati, in questo capitolo, solo problemi con equazioni lineari. Ma il computer può essere usato per risolvere equazioni molto più difficili. In effetti, è proprio in problemi che coinvolgono equazioni di secondo e terzo grado, esponenziali e simili dove il computer realmente incomincia a ripagarsi. Considerate le due seguenti equazioni simultanee:

$$2^x = (16/3)y \quad 3^x = 27\ y$$

Non è molto semplice risolvere manualmente queste due equazioni. Ma è possibile scrivere un semplice programma Basic, in grado di risolvere le equazioni per successivi tentativi. Questo è talvolta definito un approccio brutale, perché viene tentata ogni possibile combinazione di numeri compresi tra un limite inferiore ed uno superiore, finché non sia raggiunta una soluzione oppure il programma non esca dai valori stabiliti.

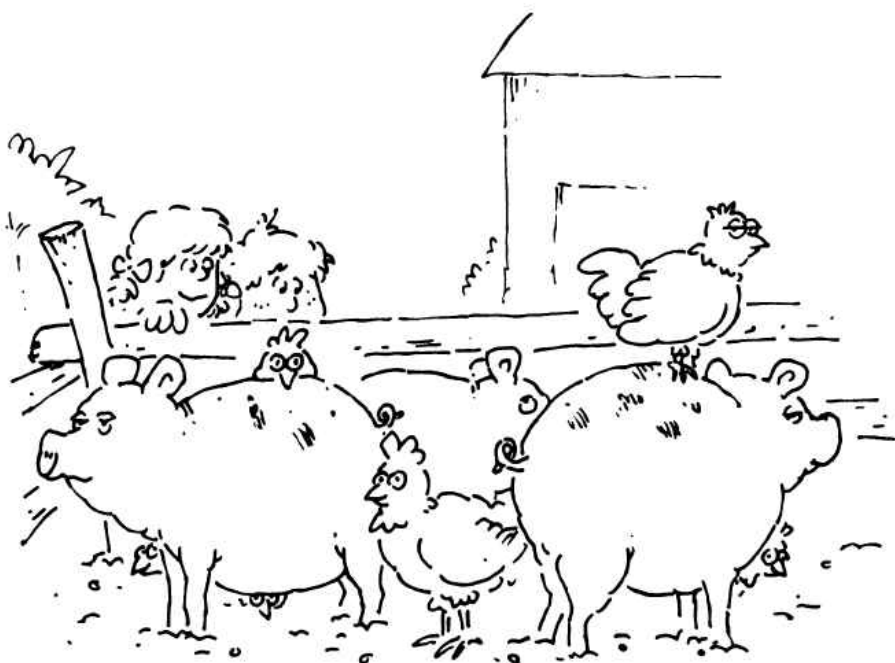
```
5 print chr$(14);"SOLUZIONE DI UN SISTEMA DI DUE EQUAZIONI"
10 for x=1 to 4
20 for y=1 to 4
30 if 2↑x<>16*y/3 or int(3↑x)<>27*y then60
40 print "x=";x,"Y=";y
50 stop
60 next y
70 next x
80 print"NESSUNA SOLUZIONE INTERA"
```

```
SOLUZIONE DI UN SISTEMA DI DUE EQUAZIONI
x= 4            Y= 3

break in 50
```

In questo caso particolare, si raggiunge piuttosto facilmente una soluzione con x = 4 e y = 3. Tuttavia, se nella seconda equazione il coefficiente y è modificato leggermente da 27 a 28, il computer tenterà 10,000 possibili soluzioni, prima di concludere che non esiste alcuna soluzione intera — almeno nel campo da 0 a 100. Attenzione: questo programma verrà eseguito in un tempo *molto* lungo.

```
30 if2↑x<>16*y/3orint(3↑x)<>28*y then 60
```

```
SOLUZIONE DI UN SISTEMA DI DUE EQUAZIONI
NESSUNA SOLUZIONE INTERA
```

L'approccio brutale per tentativi ed errori, benché sia largamente usato, è altamente inefficiente. In generale, è preferibile un approccio sistematico o di tentativi per errore guidati, a quello che tenta semplicemente ogni possibile soluzione. Comunque, per alcuni problemi il semplice approccio del tipo "tenta ogni valore" può essere appropriato. (Un'ampia trattazione degli approcci di tentativi per errore può essere trovata nelle pagine 36-40 di *Computers in Mathematics: A Sourcebook of Ideas.*
Fin qui sono stati discussi tre approcci per la soluzione di problemi. Tenendoli a mente, come risolvereste questo problema? Un ragazzo e sua sorella visitano una fattoria, dove vedono un recinto pieno di maiali e polli. Al loro ritorno a casa, il ragazzo osserva che c'erano 18 animali in tutto, mentre la sorella ricorda di aver contato un totale di 50 zampe. Quanti maiali c'erano nel recinto?

Risoluzione di equazioni quadratiche

Per qualunque valore di A, B e C di un'equazione quadratica di primo grado ($Ax^2 + Bx + C = O$), questo programma calcola le radici dell'equazione stessa. La soluzione è basata sul teorema quadratico che trova le radici con la seguente formula:

$$X = \frac{-B \pm \sqrt{B^2 - 4AC}}{2A}$$

Supposto che A, B e C siano numeri reali, sono applicabili i seguenti principi:

1. Se $B^2 - 4AC$ è positivo, allora le radici sono reali e distinte.
2. Se $B^2 - 4AC$ è uguale a 0, allora vi sono due radici reali e coincidenti.
3. Se $B^2 - 4AC$ è negativo, allora le radici sono immaginarie e distinte.

Il programma tiene in debito conto tutte queste possibilità ed identifica correttamente il tipo di radici, insieme ai loro valori, per ogni serie di coefficienti.
È utile, in sè, questo programma? Probabilmente no, eccetto che per risolvere equazioni algebriche scolastiche. Tuttavia esso potrebbe essere molto utile come routine per risolvere le equazioni quadratiche che si potrebbero incontrare in un programma più ampio.

```
5 print chr$(14)
10 print:print "EQUAZIONI QUADRATICHE"
20 input "a,b,c";a,b,c
30 r=b↑2-4*a*c
40 if a<>0 then 60
50 print "EQUAZIONE DI PRIMO GRADO":goto 150
60 print "LE RADICI SONO ";
70 if r<0 then 120
80 if r=0 then 110
90 print (-b+sqr(r))/2*a;(-b-sqr(r))/2*a
100 goto 150
110 print -b/2*a:goto 150
120 print"IMMAGINARIE"
130 print(-b/2*a);"+";sqr(-r)/2*a;"*I"
140 print(-b/2*a);"-";sqr(-r)/2*a;"*I"
150 input "UN'ALTRA (S,N)";a$
160 if a$="s" then 10
170 end
```

```
EQUAZIONI QUADRATICHE
a,b,c ?  2 , 2 , 1
LE RADICI SONO IMMAGINARIE
-2 + 2 *I
-2 - 2 *I
UN'ALTRA (S,N) ? S

EQUAZIONI QUADRATICHE
a,b,c ?  1 , 1 , 4
LE RADICI SONO IMMAGINARIE
-.5 + 1.93649167 *I
-.5 - 1.93649167 *I
UN'ALTRA (S,N) ? S

EQUAZIONI QUADRATICHE
a,b,c ?  6 , 0 , 1
LE RADICI SONO IMMAGINARIE
 0 + 14.6969385 *I
 0 - 14.6969385 *I
UN'ALTRA (S,N) ? S

EQUAZIONI QUADRATICHE
a,b,c ?  7 , 8 , 5
LE RADICI SONO IMMAGINARIE
-28 + 30.5122926 *I
-28 - 30.5122926 *I
UN'ALTRA (S,N) ? S


EQUAZIONI QUADRATICHE
a,b,c ?  8 , 0 , 1
LE RADICI SONO IMMAGINARIE
 0 + 22.627417 *I
 0 - 22.627417 *I
UN'ALTRA (S,N) ? N
```

Risoluzione di equazioni esponenziali

Un'altra routine di uso generale per risolvere un tipo particolare di equazione è la seguente, in grado di trovare l'esponente in un'equazione esponenziale. Dati i valori A, B, m e n, questo programma determina x in ogni equazione esponenziale nella forma:

$$A^{mx+n} = B$$

Per esempio, il programma risolverà tutti i seguenti problemi:

1. $5^X = 40$
2. $5^{3X+1} = 7.6$
3. $17^{X-3} = 8.12$
4. $11^{1-2X} = 247$

Dovendo risolvere manualmente un'equazione esponenziale, seguireste probabilmente i seguenti passi:

$$5^X = 40$$
$$\log 5^X = \log 40$$
$$x \log 5 = \log 40$$
$$x = \log 40/\log 5$$
$$x = 1.6021/.6990 = 2.292$$

Tuttavia, in forma più generalizzata, la soluzione è:

$$X = \frac{\frac{\log B}{\log A}}{M-(N/M)}$$

I dati per i quattro problemi prima indicati sono stati inseriti nelle quattro istruzioni DATA (linee 100 — 130).
Il programma qui presentato può essere migliorato in vari modi. Primo, risolve sempre le stesse quattro equazioni. Potete generalizzarlo in modo che risolva altre equazioni? Secondo, se doveste utilizzare questa routine in un altro programma probabilmente non potreste usare istruzioni READ; come si può affrontare questo inconveniente?

```
5 print chr$(14)
10 print chr$(147);"SOLUTORE EQUAZIONI ESPONENZIALI"
20 print
30 print " A       B       M       N       X"
40 print "---     ---     ---     ---     ---"
50 read a,b,m,n
60 x=(log(b)/log(a))/m-(n/m)
70 x=int(.5+100*x)/100
80 print a;tab(7);b;tab(14);m;tab(21);n;tab(27);x
90 goto 50
100 data 5,40,1,0
110 data 5,7.6,3,1
120 data 17,8.12,1,-3
130 data 11,247,-2,1
```

```
SOLUTORE EQUAZIONI ESPONENZIALI

 A       B       M       N       X
---     ---     ---     ---     ---
 5      40      1       0      2.29
 5      7.6     3       1       .09
 17      8.12     1      -3       3.74
 11      247     -2       1      -.65
```

Radici di una funzione qualunque

Questo programma trova le radici di una funzione, di qualunque funzione! La funzione può essere lineare, quadratica, cubica, trigonometrica, o qualunque combinazione di esse, lunga quanto può essere rappresentata nel linguaggio Basic. Il programma, così come è qui descritto, trova le radici comprese tra —20 e +20, tuttavia voi potete modificare questi limiti intervenendo sulla istruzione 50.
Il metodo utilizzato comporta il calcolo della funzione a piccoli intervalli incrementali, l'individuazione dei punti in cui la funzione cambia segno e poi, per successive approssimazioni, l'individuazione del punto in cui si azzera. Questo approccio deriva dal metodo di Newton nella parte finale ma, a differenza di questo, funziona anche se si effettua un'ipotesi iniziale poco felice.
Prima di eseguire il programma, dovete scrivere la vostra funzione nella linea 30. Per esempio:

```
DEF FNA (X) = 2*X↑3+11*X↑2-31*X-180
DEF FNA (X) = X-4
DEF FNA (X) = SIN(X)-.5
```

Potrà rivelarsi necessario far riferimento al manuale Basic del vostro sistema per verificare esattamente come si debba definire una funzione.
La routine in questo programma è molto potente e potrebbe essere usata come subroutine in molti altri programmi.
Come potreste utilizzare questo programma per risolvere il seguente problema?

$$x = \sqrt{12+\sqrt{12+\sqrt{12+\sqrt{12+\sqrt{12+\ldots}}}}}$$

```
5 print chr$(14):rem setta minuscole
10 print chr$(147);"Radici di qualsiasi funzione."
20 print "Funzione definita nella linea 30"
30 def fna(x)=2*x↑3+11*x↑2-31*x-180
40 z1=-200
50 fori=-19.9 to 20
60 if sgn(fna(i))=sgn(fna(i+1)) then 230
70 k=i
80 j=i+1
90 if fna(k)<fna(j) then 130
100 z=k
110 k=j
120 j=z
130 z=(k+j)/2
140 if fna(z)<0 then 170
150 j=z
160 goto 180
170 k=z
180 if abs(fna(z))>.00005 then 130
190 z=sgn(z)*int(abs(z)*10000+.05)/10000
200 if z=z1 then 230
210 print "f(";z;")=0"
220 z1=z
230 next i
```

```
Radici di qualsiasi funzione.
Funzione definita nella linea 30
f(-5 )=0
f(-4.5 )=0
f( 4 )=0
```

```
30 def fna(x)=x-4
```

```
Radici di qualsiasi funzione.
Funzione definita nella linea 30
f( 4 )=0
```

```
Radici di qualsiasi funzione.
Funzione definita nella linea 30
f(-18.3259 )=0
f(-16.2315 )=0
f(-12.0428 )=0
f(-9.9483 )=0
f(-5.7596 )=0
f(-3.6651 )=0
f( .5235 )=0
f( 2.6179 )=0
f( 6.8067 )=0
f( 8.9011 )=0
f( 13.0899 )=0
f( 15.1843 )=0
f( 19.3731 )=0
```

```
30 def fna(x)=sin(x)-.5
```

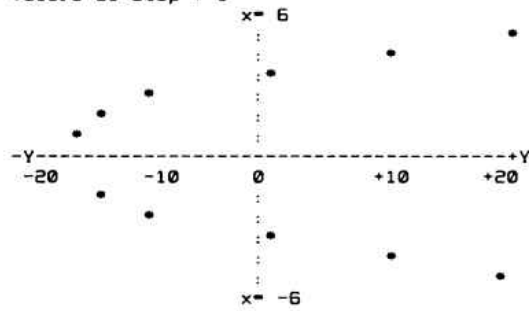
Tracciamento della curva di una funzione qualsiasi

Ecco ora un programma che genera su schermo o su stampante la curva di una qualunque funzione.
Prima di eseguire il programma, dovete scrivere la vostra funzione nella linea 200. Come il precedente programma trovava le radici di una funzione qualunque, questo traccia la curva di una qualunque funzione. Dovete dire al programma tra quali valori volete che la funzione sia disegnata, cioè comunicare un valore minimo ed uno massimo sull'asse delle X, così come anche l'incremento desiderato.
Il programma, così come è qui riportato, non consente la selezione di valori sull'asse delle Y; il programma traccia infatti valori compresi tra —30 e +30.
Ecco ora varie funzioni che potreste provare e tracciare:

Funzione	*Limiti su X*		*Incremento*
DEF FNA(X) = 2•X	-15	15	1
DEF FNA(X) = 30•SIN(X)	-5	5	.25
DEF FNA(X) = X-X•X	-5	6	.5
DEF FNA(X) = 30•EXP(-X•X/100)	-30	30	1.5
DEF FNA(X) = X•X-X	-5	6	1
DEF FNA(X) = X↑2-X-15			

L'ultima funzione è quella tracciata nell'esempio qui illustrato. Le funzioni esponenziali sono molto divertenti e talvolta conducono a risultati inaspettati ed interessanti, in particolare quando vengono combinate con funzioni trigonometriche. Provate e divertitevi!

```
5 print chr$(14):rem setta minuscole
10 print chr$(147);"Disegna qualsiasi funzione"
20 print "Funzione definita nella  Linea 30"
30 def fna(x)=x↑2-15
40 input "Valore iniziale di x";x1
50 input "Valore finale di x";x2
60 input "Valore di step";s
70 print tab(19);"x=";x1
80 for x=x1 to x2 step -s
90 if abs(x)<.00001 then 160
100 y=fna(x)+20
110 if y<40 then 130
120 y=39
130 if y>20 then 200
140 print tab(y);"•";tab(20);":"
150 goto 210
160 print "-y";:for t=1 to 36:print "-";:next t: print "+y"
170 print " -20       -10       0       +10       +20"
180 x=x-s
190 goto 210
200 print tab(20);":";tab(y);"•"
210 next x
220 print tab(19);"x=";x2
230 goto 230
```

Disegna qualsiasi funzione
Funzione definita nella linea 30
Valore iniziale di x ? 6
Valore finale di x ? -6
Valore di step ? 1
x= 6
-Y
+Y
-20
-10
0
+10
+20
x= -6

3
Serie e prove ripetitive

Per la soluzione di problemi relativamente semplici, il computer può non essere di alcun aiuto. Infatti, si può impiegare più tempo a scrivere un programma per risolvere un problema, che non a risolverlo a mano o con una calcolatrice. Questo capitolo dovrebbe aiutarvi a riconoscere quali problemi siano adatti al computer e quali no. In alcuni problemi si può pensare che il computer non sia di alcun aiuto. Tuttavia, scrivere un programma per computer vi forzerà sempre a ragionare su un approccio in grado di risolvere il problema in modo logico e preciso. Il computer non può risolvere problemi se non gli viene detto esattamente come procedere; perciò è necessario che abbiate compreso completamente un problema prima che possiate tradurlo in un programma per computer.

Molte sezioni di questo capitolo trattano le serie di dati. Benché le serie usate in questi esempi contengano un numero relativamente piccolo di elementi o valori, è bene che teniate a mente che i problemi del mondo reale comportano l'analisi di migliaia o milioni di dati e che l'unico modo pratico per risolvere problemi di queste dimensioni è tramite un computer. Per esempio, pensate a come potreste ottimizzare la spedizione per ferrovia di 4000 diversi carichi diretti da Boston a Phoenix e ad altri 780 punti intermedi. Considerate poi che vi sono 200 treni merci in partenza ogni giorno da Boston. A ciò aggiungete gli altri 10.000 treni che partono ogni giorno da altre città e che devono utilizzare la stessa rete ferroviaria: potete allora vedere che non è un compito facile avere a che fare con insiemi di dati del mondo reale.

Gruppi di ragazze e ragazzi

Nel capitolo precedente abbiamo detto che ci possono essere alcuni problemi per i quali è appropriato il metodo brutale per tentativi ed errori. È il caso di problemi relativamente piccoli, per i quali tentare ogni possibile soluzione non comporta un grande spreco di considerevole tempo macchina. Ecco un problema, riguardante due equazioni lineari, che conduce ad un approccio del tipo per tentativi ed errori.
Il problema è il seguente: 15 ragazze abbandonano un gruppo di ragazzi e ragazze, lasciando così due ragazzi per ogni ragazza (fortuna per le ragazze!). Poi, 45 ragazzi decidono di andarsene; ci sono adesso 5 ragazze per ogni ragazzo (fortuna per i ragazzi!). Quante ragazze c'erano all'inizio del gruppo?
Prima di correre verso il computer, dovete riconoscere che questo problema richiede la soluzione di due equazioni simultanee. Se G è il numero iniziale di ragazze e B il numero iniziale di ragazzi, allora le due equazioni sono:

$$(G - 15) \times 2 = B$$

$$(B - 45) \times 5 = (G - 15)$$

```
5 print chr$(14):rem setta minuscole
10 i=0
20 print "Il programma e' lento, sii paziente...."
30 for g=1 to 100
40 for b=1 to 100
50 i=i+1
60 if 2*(g-15)<>b then 100
70 if 5*(b-45)<>(g-15) then 100
80 print g;"ragazze   ";b;"ragazzi"
90 goto 130
100 next b
110 next g
120 print "Nessuna soluzione intera da 0 a 100."
130 print i;"prove."
```

```
Il programma e' lento, sii paziente....
 40 ragazze    50 ragazzi
 3950 prove.
```

Il programma utilizza due loop FOR (istruzioni 30-110 e 40-100) per provare ogni combinazione di valori per B e G compresi tra 1 e 100, finché non sia trovata una soluzione oppure il programma non esca dai valori stabiliti. La variabile I (linea 50) è un contatore che registra il numero di tentativi richiesti per raggiungere una soluzione.

Il programma è semplice e lineare, e trova una soluzione dopo 3950 tentativi. Sarebbe però stato semplice sostituire nella seconda equazione il valore di B derivante dalla prima e risolvere rapidamente il problema a mano o con una calcolatrice. È importante riconoscere che se un problema può essere risolto facilmente con altri metodi, il computer offre un vantaggio scarso o nullo.
Provate questo problema, per il quale potete voler usare un computer, oppure no. Se Matthew batte Jeff di un decimo di miglio in una gara da due miglia e Jeff può battere Steven di un quinto di miglio in una gara da due miglia, con quale distacco Matthew può battere Steven in una gara da due miglia? (Un suggerimento: la risposta non è 3/10 di miglio).

I libri di Brawn

L'uso dell'approccio per tentativi ed errori può in genere essere significativamente ottimizzato se le combinazioni da tentare possono essere in qualche modo ristrette. La soluzione a questo problema mostra come la velocità nel fornire una soluzione possa essere aumentata di oltre 100 volte combinando fra loro le equazioni ed eliminando certe possibili soluzioni.
Ed ecco il problema. Brawn vende 48 libri ad un mercatino delle pulci, alcuni per $ 3 ciascuno, altri per $ 5 ed altri ancora per $ 8, raccogliendo in totale $ 75. Egli ricorda di aver avuto un numero pari di libri da $ 5. Siete in grado di determinare quanti libri aveva per ogni tipo?
Le equazioni risolutrici sono (ponendo T uguale al numero di libri da $ 3, F al numero di libri da $ 5, ed E al numero di libri da $ 8):

$$T + F + E = 48 \qquad 3*T + 5*F + 8*E = 175$$

Il primo programma è stato scritto semplicemente per provare tutte le possibili combinazioni di T, F ed E da 1 a 48. Esso porta tre soluzioni del problema; le due soluzioni con un numero dispari di libri da $ 5 possono essere eliminate, lasciando così soltanto l'unica soluzione desiderata.

```
5 print chr$(14):rem setta minuscole
10 print "I libri di Brawn."
20 print "Tempo lungo di elaborazione"
40 print " $3    $5    $8"
50 for t=1 to 58
60 for f=1 to 48
70 for e=1 to 48
80 if (t+f+e)<>48 then 110
90 if t*3+f*5+e*8<>175 then 110
100 print t;tab(6);f;tab(13);e
110 next e
120 next f
130 next t
```

```
I libri di Brawn.
Tempo lungo di elaborazione
$3    $5    $8
40    3     5
37    8     3
34    13    1
```

Questo programma richiede circa 22 minuti e 13 secondi per girare su un Commodore 64. Un tipico minicomputer (PDP-8/e) può risolvere lo stesso problema in circa 7.3 secondi. In entrambi i casi, si tratta di un lungo impiego di tempo macchina.
È piuttosto facile combinare le due equazioni in una determinando l'incognita T. L'unica equazione è allora:

$$2*F + 5*E = 31$$

In questa equazione, i limiti possono essere ridotti (rispetto ai 48 usati la prima volta) poiché F non può essere maggiore di 31/12 o 15.5 ed E non può essere maggiore di 31/5 o 6.2. Apportando le appropriate modifiche al programma si ottiene il seguente secondo programma.

```
5 print chr$(14):rem setta minuscole
10 print "I libri di Brawn."
40 print " $3     $5     $8"
60 forf=1to15
70 fore=1to6
80 if(f*2+e*5)<>31then110
90 t=48-f-e
100 print t;tab(6);f;tab(13);e
110 next e
120 next f

I libri di Brawn.
Tempo lungo di elaborazione
$3     $5     $8

40     3      5
37     8      3
34     13     1
```

L'uso di questo programma comporta una drastica riduzione del tempo richiesto per ottenere la soluzione. Sul Commodore 64, il tempo si riduce a circa 2.5 secondi e sul PDP-8 a circa 0.16 secondi.

Poiché il problema dice che F deve essere pari, si può apportare una modifica finale alla linea 20, che incrementa F di 2 unità alla volta. Questa versione richiede solo 1.25 secondi sul Commodore 64 e 0.06 secondi sul PDP-8.

```
5 print chr$(14):rem setta minuscole
10 print "I libri di Brawn."
30 print"    F    Dispari"
40 print " $3     $5     $8"
60 forf=2to14step2
70 fore=1to6
80 if(f*2+e*5)<>31then110
90 t=48-f-e
100 print t;tab(6);f;tab(13);e
110 next e
120 next f

I libri di Brawn.
Tempo lungo di elaborazione
   F    Dispari
$3     $5     $8
37     8      3
```

Osservate l'enorme riduzione del tempo richiesto per una soluzione, di oltre 1000 volte sul Commodore 64 e 100 volte sul PDP-8. L'approccio brutale è certamente inefficiente! Vale in genere la pena di riflettere su molti problemi, in particolare di grosse dimensioni, prima di precipitarsi ad affrontarli sul computer. Il computer sarà anche veloce, ma noi abbiamo migliorato le sue prestazioni di 1000 volte usando semplicemente un po' di buon senso.

Intersezioni di insiemi

Due insiemi di numeri possono essere uniti, ottenendo un terzo insieme, con l'operazione di intersezione. L'intersezione di due insiemi A e B è l'insieme che contiene tutti gli elementi che appartengono sia ad A che a B, senza contenere alcun altro elemento. L'intersezione è solitamente scritta nella forma A ∩ B.
Per esempio, se M = <0,2,4,6> e K = <1,2,3,4), allora M ∩ K = <2,4>.
Questo programma trova l'intersezione di due insiemi di numeri. È stato scritto per trovare l'intersezione di due serie ripetitive descritte nelle istruzioni 30 e 40. Nell'esempio, la linea 30 descrive l'insieme x = <1,3,5,...19>, mentre la linea 40 descrive l'insieme y = <2,5,8,...29>.
Osservate che i valori di x si incrementano di 2 e quelli di y di 3.

```
10 print chr$(14):rem setta minuscole
20 print "Gli insiemi x e y si intersecano nei    punti:";
30 for x=1 to 19 step 2
40 for y=2 to 29 step 3
50 if x=y then 90
60 next y
70 next x
80 end
90 print x;
100 goto 70
```

```
Gli insiemi x e y si intersecano nei    punti: 5  11  17
```

Tuttavia, se gli insiemi non possono essere descritti così esattamente, può essere auspicabile riscrivere il programma in modo tale che esso esamini ogni possibile insieme di dati. Ciò si ottiene con l'istruzione READ che legge in x i dati contenuti nell'istruzione DATA. Il programma è impostato in modo tale da usare lo stesso insieme y del primo programma, mentre l'insieme x è definito nell'istruzione DATA.
Dalla prima combinazione di insiemi dovreste essere in grado di vedere che se gli insiemi intersecantisi possiedono una caratteristica numerica, allora anche l'insieme risultante avrà una caratteristica. Nell'esempio, i valori di x si incrementano di 2 e quelli di y di 3, quindi i valori nell'insieme derivante dalla loro intersezione si incrementano di 2x3=6. L'intersezione di questi insiemi può essere facilmente calcolata a mano, ma il computer può costituire un aiuto nel calcolo su insiemi più complicati.

```
5 print chr$(14):rem setta minuscole
10 print "Gli insiemi x e y si intersecano nei     punti:";
20 read x
30 for y=2 to 29 step 3
40 if x=y then 80
50 next y
60 goto 20
70 end
80 print x;
90 goto 60
100 data 2,3,8,9,14,15,20,21,26,27
```

```
Gli insiemi x e y si intersecano nei    punti: 2  8  14  20  26
```

Numeri primi

Un numero primo è un intero positivo divisibile solo per se stesso e per 1. I primi dieci numeri primi sono 2, 3, 5, 7, 11, 13, 17, 19, 23 e 29. La definizione dà il metodo di base per determinare se un numero è primo: dividerlo per tutti i numeri interi minori di esso fino a 2, controllare se il resto è zero per almeno uno di loro. Se non è così, allora il numero è primo.
Ma questo metodo è estremamente inefficiente. È ovvio che un numero non è primo se è pari e maggiore di 2; vanno perciò cercati solo divisori dispari. Inoltre, non è necessario cercare divisori maggiori della radice quadrata del numero.
Poiché il metodo della divisione è inefficiente, sono stati individuati vari schemi per evitare la divisione. L'idea che sta alla base di tutti questi schemi è chiamata crivello di Eratostene (276 a.C.-195 a.C.). Pensate ad una lista di numeri dispari dal 3 in su. Eliminate ogni terzo numero dopo 3, ogni quinto numero dopo 5, e così via. Otterrete così solo numeri primi.

```
5 print chr$(14):rem setta minuscole
10 dim a(100)
20 c=0
30 print "Numeri primi"
40 print "Digitare 0 per terminare"
50 input "Il tuo numero";m
60 if m=0 then end
70 n=m
80 x=0
90 if m>0 then 110
100 print "Non accettabile":goto 50
110 i=1
120 i=i+1
130 if i>m then 190
140 if m/i<>int(m/i) then 120
150 x=x+1
160 a(x)=i
170 m=m/i
180 goto 140
190 if x=1 then 240
200 for i=1 to x
210 print a(i);
220 next i
230 goto 260
240 print n;"e' un numero primo."
260 print
270 goto 50
280 end
```

```
Numeri primi
Digitare 0 per terminare
Il tuo numero ?  65
 5  13
Il tuo numero ?  118
 2  59
Il tuo numero ?  55
 5  11
Il tuo numero ?  131
 131 e' un numero primo.

Il tuo numero ?  93
 3  31
Il tuo numero ?  77
 7  11
Il tuo numero ?  143
 11  13
Il tuo numero ?  119
 7  17
Il tuo numero ?  111
 3  37
Il tuo numero ?  104
 2  2  2  13
Il tuo numero ?  18
 2  3  3
Il tuo numero ?  35
 5  7
Il tuo numero ?  9
 3  3
Il tuo numero ?  117
 3  3  13
Il tuo numero ?  76
 2  2  19
Il tuo numero ?  104
 2  2  2  13
```

Questo programma trova la suddivisione in fattori primi di qualunque intero, oppure scrive "N è primo" se l'intero non ha divisori.
Fate girare questo programma con una gran quantità di numeri e vedete se siete in grado di scoprire delle relazioni tra i numeri ed i loro fattori primi. Dovreste anche tentare di immaginare quale metodo è stato usato nel programma per trovare i fattori primi di qualunque numero intero. Per fare ciò potreste disegnare un flow-chart che illustri ciò che avviene nel programma. Questo vi aiuterà ad individuare il metodo usato per trovare un numero primo e vi aiuterà a scrivere un programma che generi i numeri primi.
Nello scrivere un programma per generare i fattori primi potete usare il metodo del crivello. Tuttavia, al crescere dei valori, dovrete immaginare un modo per rappresentare numeri interi con più cifre di quante il vostro computer può trattare. (Un possibile approccio è descritto alle pagine 19-21 di *Computers in Mathematics*).
Il matematico Goldbach ipotizzò che ogni numero pari maggiore di 4 può essere espresso come la somma di due numeri primi ($16 = 11 + 5$, $30 = 17 + 13$, ecc.). Nessuno è mai riuscito a provarlo, né a dimostrare che è errato. Ecco perché è definito una congettura. Siete in grado di scrivere un programma che provi la correttezza o l'inesattezza di questa congettura? E che ne dite di scrivere un programma che dimostri l'ipotesi di Goldbach per i numeri pari fino a 50? Dovreste riuscire a scrivere questo programma con non più di 12 istruzioni.
Ecco un altro problema riguardante i numeri interi. Ipotizzando una durata della vita di 80 anni, in quale anno del 20° secolo (1900-1999) dovrebbe essere nata una persona per avere il maggior numero di compleanni che siano numeri primi? Ed il minor numero?

Massimo comune divisore

Il massimo comune divisore di un insieme di numeri è, come dice il suo nome, il più grande numero intero per cui ogni numero dell'insieme possa essere diviso. Per esempio, l'insieme di numeri 12, 20 e 28 ha 4 come massimo comune divisore. Nessun altro valore maggiore di 4 può infatti essere divisore di tutti e tre i numeri. Questo programma trova il massimo comune divisore per qualunque insieme di numeri interi. Per esempio, dovete semplicemente inserire la quantità di numeri interi nel vostro insieme, scriverli quando richiesto e far calcolare al programma il MCD. Il nucleo centrale del calcolo si trova nell'istruzione 150.

```
5 print chr$(14):rem setta minuscole
10 dim x(50)
20 print chr$(147)
30 print "Massimo Comune Divisore"
35 print
40 input "Quanti numeri";n
50 print "Numeri";
60 rem ????????????????
70 for k=1 to n
80 input x(k)
90 if x(k)>s then 110
100 s=x(k)
110 next k
120 g=0
130 for m=2 to 5
140 for i=1 to n
150 if x(i)/m<>int(x(i)/m) then 180
160 next i
170 g=m
180 next m
200 if g>0 then 230
210 print "Sono relativamente primi."
220 goto 240
230 print "Il MCD e'";g
240 input "Riproviamo (s,n)";z$
250 if z$="s" then 35
260 print "Okay. Ciao"
```

```
Massimo Comune Divisore

Quanti numeri ?  0
Numeri ?  18
Il MCD e' 3
Riproviamo (s,n) ?

Quanti numeri ?  5
Numeri ?  32
 25
 32
 33
 25
Sono relativamente primi.
Riproviamo (s,n) ?

Quanti numeri ?  5
Numeri ?  31
 5
 20
 0
 35
Sono relativamente primi.
Riproviamo (s,n) ?

Quanti numeri ?  3
Numeri ?  14
 28
 46
Il MCD e' 2
Riproviamo (s,n) ?

Quanti numeri ?  5
Numeri ?  41
 43
 38
 34
 18
Sono relativamente primi.
Riproviamo (s,n) ?

Quanti numeri ?  2
Numeri ?  49
 11
Sono relativamente primi.
Riproviamo (s,n) ?
```

Conoscete il significato di un insieme di numeri relativamente primo? Potete immaginarne il significato dal terzo esempio del programma o da esempi fatti da voi? In che consiste la differenza tra un insieme di numeri relativamente primi e un insieme di fattori primi? Riuscite a trovare un insieme relativamente primo di 10 numeri interi?

Nel paragrafo precedente abbiamo discusso dei numeri primi. Eccovi una sfida interessante riguardante i numeri primi. Fino alla fine del 1982, la più lunga progressione di numeri interi in cui ognuno si distanzia dal precedente dello stesso valore era 17. Il Prof. Paul Pritchard, del dipartimento di informatica della Cornell University ha scritto un programma in grado di determinare se ci fosse una progressione più lunga. Usando un DEC VAX 11/780 trovò la serie di 18 numeri mostrata qui sotto. Scoprì anche altre 14 progressioni di 17 numeri e 10 da 18 numeri, ma nessuna con 19 numeri. Egli è convinto che ve ne sia almeno una; siete capaci di trovarla?

107928278317	197233324147
117851061187	207156107017
127773844057	217078889887
137696626927	227001672757
147619409797	236924455627
157542192667	246847238497
167464975537	256770021367
177387758407	266692804237
187310541277	276615587107

Problemi crittografici

I problemi crittografici (o criptaritmetici, o alfametici) sono espressioni aritmetiche in cui le cifre sono sostituite da lettere dell'alfabeto. Ogni cifra è associata ad una lettera, in modo da ottenere una frase interessante, come per esempio:

```
    SEND
  + MORE
  ------
   MONEY (*)
```

Se lo studente che ha inviato questo messaggio al padre aveva bisogno di $ 106.52 per pagarsi il biglietto d'aereo per tornare a casa, questo era proprio il messaggio giusto da inviare poiché questa combinazione di lettere ha una sola possibile soluzione:

```
    9567
  + 1085
  ------
   10652
```

Tuttavia, se lo studente avesse atteso l'ultimo momento e fosse di gran fretta, avrebbe potuto scrivere il messaggio in quest'altro modo:

```
    WIRE
  + MORE
  ------
   MONEY
```

In questo caso, quanto denaro avrebbe dovuto inviare il padre? Abbiamo precedentemente discusso gli approcci per tentativi ed errori nella soluzione di problemi. Abbiamo anche notato che l'approccio brutale consistente nel tentare ogni possibile alternativa è talvolta appropriato. È questo un caso del genere?
No! Il numero di possibili alternative è dato dal fattoriale del numero di lettere differenti presenti nell'espressione alfametica, ovvero 8! o 40.320. Un programma che tentasse tutte queste possibilità girerebbe per un tempo molto l-u-n-g-o.
In questo caso è molto più efficiente applicare un po' di buon senso per restringere il numero di alternative. L'approccio migliore consiste nel suddividere lo spazio di ricerca in ampie classi (o insiemi), in base a qualche proprietà comune condivisa dai membri di ogni classe e poi cercare di eliminare intere classi con il metodo della contraddizione.
Consideriamo l'ultimo problema posto. Può essere $E = O$? Poiché $E + E = Y$, anche Y sarebbe O, contraddicendo così il fatto che E ed Y devono corrispondere a cifre diverse. Perciò l'intera classe di soluzioni in cui $E = O$ può essere eliminata.

(*) È evidentemente impossibile tradurre in italiano queste operazioni crittografiche, mantenendone la validità aritmetica. Il significato letterale delle due operazioni riportate su questa pagina è rispettivamente: MANDAMI DENARO e INVIAMI (per telegrafo) ALTRO DENARO.
Potrebbe essere un interessante esercizio, per il lettore italiano, creare analoghe operazioni crittografiche, utilizzando vocaboli italiani (N.d.T.).

Consideriamo ora E = 3. Allora Y = 6 e non c'è riporto alla colonna successiva. Perciò in tale colonna R + R = E, oppure E + 10 se c'è un riporto. Ma in entrambi i casi E dovrà essere pari, essendo 2R un numero sempre pari; ciò contraddice l'ipotesi che E = 3.
Seguendo questo tipo di elaborazione di classi di contraddizione, per ogni cifra nell'ordine E,R,I,O e N, il computer individuerà tutte le possibili soluzioni del problema. Il secondo problema, a differenza del primo, ha cinque possibili soluzioni. Un padre in gamba avrebbe scelto la soluzione più bassa, inviando al figlio $ 103.48.

```
5 print chr$(14):rem setta minuscole
10 print chr$(147)
20 print "Solutore di criptogrammi"
30 print
40 print "  W I R E"
50 print " +M O R E"
60 print "---------"
70 print "M O N E Y"
90 print
100 m=1
110 for e=2 to 9
120 y=e+e
130 if y>10 then 160
140 c1=0
150 goto 180
160 c1=1
170 y=y-10
180 for r=0 to 9
190 if r=m or r=e or r=y then 480
200 if r+r+c1=e then 230
210 if r+r+c1=e+10 then 250
220 goto 480
230 c2=0
240 goto 260
250 c2=1
260 for i=1 to 9
270 if i=m or i=e or i=y or i=r then 470
280 for o=0 to 9
290 if o=m or o=e or o=y or o=r or o=i then 460
300 n=i+o+c2
310 if n>=10 then 340
320 c3=0
330 goto 360
340 c3=1
350 n=n-10
360 if n=m or n=e or n=y or n=r or n=i or n=o then 460
370 for w=0 to 9
380 if w=m or w=e or w=y or w=r or w=i or w=o or w=n then 450
390 if o+10<>w+m+c3 then 450
400 print
410 print "   ";w;i;r;e
420 print " + ";m;o;r;e
430 print " -------------"
440 print m;o;n;e;y
445 print:print
450 next w
460 next o
470 next i
480 next r
490 next e
500 end
```

```
Solutore di criptogrammi
                                        9  5  7  4
  W I R E                            +  1  0  7  4
 +M O R E                            -------------
 ---------                           1  0  6  4  8
 M O N E Y

    9  7  6  2                          9  2  8  7
 +  1  0  6  2                       +  1  0  8  7
 -------------                       -------------
 1  0  8  2  4                       1  0  3  7  4


    9  2  7  4                          9  5  8  7
 +  1  0  7  4                       +  1  0  8  7
 -------------                       -------------
 1  0  3  4  8                       1  0  6  7  4
```

Vi sono altri approcci alla soluzione dei problemi crittografici, ma tutti traggono grande beneficio dal fatto di ridurre al minimo possibile lo spazio di ricerca, prima di impostare il problema sul computer. Vedete se siete in grado di individuare un altro efficace approccio e scrivere un programma che lo implementi.
Ed eccovi qualche altro problema da provare a risolvere. (*)

```
  DONALD        TWO        ABCDE
+ GERALD      x TWO          x 4
--------      -----        -----
  ROBERT      THREE        EDCBA

    ABC         THE       SPRING
   x DE       EARTH        RAINS
   ----       VENUS        BRING
    FEC      SATURN      + GREEN
   DEC     + URANUS      -------
   ----    --------       PLAINS
   HGBC     NEPTUNE

    ONE       FORTY         FIVE
    TWO         TEN        -FOUR
 + FIVE       + TEN        -----
 ------       -----          ONE
  EIGHT       SIXTY        + ONE
                           -----
                             TWO
```

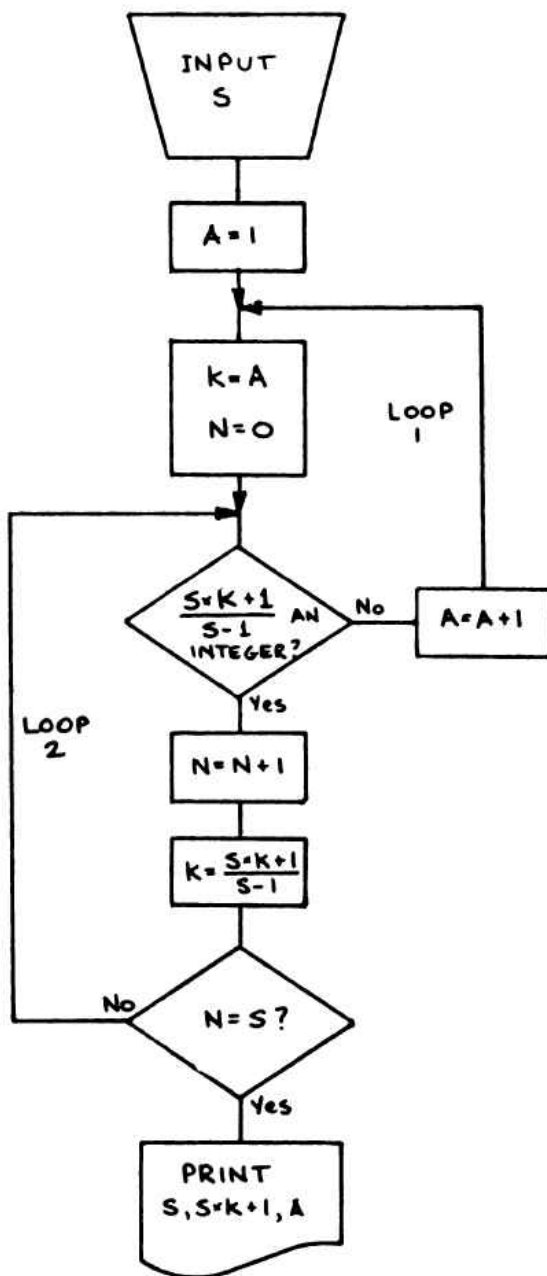
VIOLIN + VIOLIN + VIOLA + CELLO = QUARTET
THREE + NINE = EIGHT + FOUR

(*) Diamo qui la traduzione letterale dei problemi esposti, nell'ordine da sinistra a destra e dall'alto in basso, DONALD + GERALD = ROBERT; DUE x DUE = TRE; ABCDE x QUATTRO = EDCBA; ABC x DE = HGBC; LA TERRA + SATURNO + VENERE + URANIO = NETTUNO; LE PIOGGE DI PRIMAVERA FANNO VERDI LE PIANURE; UNO + DUE + CINQUE = OTTO; QUARANTA + DIECI + DIECI = SESSANTA; CINQUE — QUATTRO = UNO + UNO × DUE; VIOLINO + VIOLINO + VIOLA + VIOLONCELLO = QUARTETTO; TRE + NOVE = OTTO + QUATTRO. (N.d.T.).

Il problema dei marinai e della scimmia

Vi presentiamo ora una tra le tante variazioni sul problema dei marinai e della scimmia.
Su un'isola vi sono cinque marinai ed una scimmia. Una sera i marinai radunano tutte le noci di cocco che riescono a trovare e le accatastano in un grosso mucchio. Essendo esausti dopo questo duro lavoro, decidono di attendere la mattina successiva per dividerle equamente. Durante la notte, un marinaio si sveglia e separa le noci in cinque mucchi uguali, ma lascia da parte una noce di cocco, che dà alla scimmia. Poi prende un mucchio, lo nasconde e raduna nuovamente insieme gli altri quattro e ritorna infine a dormire. Successivamente la stessa azione viene eseguita dagli altri quattro marinai, ognuno dei quali fa esattamente le stesse operazioni. La mattina successiva le noci rimaste vengono equamente divise, lasciandone una, che avanza, alla scimmia. Qual è il numero minimo di noci di cocco con cui essi hanno iniziato?
Benchè ci sia un'elegante soluzione algebrica al problema, un approccio più adatto al computer è quello del processo retroattivo. Una tipica soluzione di un problema può essere pensata come un percorso che conduce dall'informazione data fino all'obiettivo desiderato. Tuttavia, questa volta l'obiettivo, o stato finale, è noto, ed è allora più facile partire da lì e procedere all'indietro fino allo stato iniziale.
Come detto pocanzi, esiste una quantità infinita di variazioni al problema. Per esempio, i marinai potrebbero essere 3 o 6 o 14 invece che 5. È perciò desiderabile individuare una soluzione generale, anziché una apposita per uno specifico problema.

Nel flow-chart e nel programma, S è il numero di marinai mentre A è la quantità di noci di cocco che ogni marinaio riceve durante la divisione finale. Poiché ad ogni divisione una noce di cocco è stata data alla scimmia, la quantità complessiva di noci rimaste al mattino deve essere S x A + 1. Ma questo mucchio deriva dall'accumulo di S — 1 mucchi uguali. Quindi la condizione chiave è che (S x A + 1)/(S—1) deve essere un numero intero K, che rappresenta la quantità di noci rubate dall'ultimo marinaio da un mucchio di SxK+1 noci. Ma questo mucchio è il risultato dell'accumulo di S—1 mucchi uguali, effettuato dal ladro precedente, perciò (SxK+1)/(S—1) è ancora un intero e così via all'indietro per tutti gli S attacchi al mucchio.
Osservate che nel flow-chart e nel programma il primo valore di prova per A è 1 (istruzione 40). Nell'istruzione 80 questo valore viene incrementato di 1 finché (SxK+1)/(S—1) non sia intero, come verificato nell'istruzione 70. Questa elaborazione continua finché il contatore del secondo ciclo, N (pari al numero di divisioni notturne) uguaglia il numero di marinai.
Il programma funziona per qualunque numero di marinai, ma impiega un notevole tempo per un numero maggiore di 5. Ricordando quanto avete precedentemente appreso in questo capitolo, siete in grado di individuare un modo per rendere più efficiente il programma?

INPUT
S
A=1
K=A
N=0
LOOP
1
S*K+1
S-1
AN
INTEGER?
No
A=A+1
Yes
LOOP
2
N=N+1
K=S*K+1
S-1
No
N=S?
Yes
PRINT
S,S*K+1,A

```
5 print chr$(14):rem setta minuscole
10 print chr$(147)
20 print "I marinai e la scimmia"
30 input "N. marinai";s
40 a=1
50 k=a
60 n=0
70 if (s*k+1)/(s-1)=int((s*k+1)/(s-1)) then 100
80 a=a+1
90 goto 50
100 n=n+1
110 k=(s*k+1)/(s-1)
120 if n=s then 140
130 goto 70
140 print "Le noci sono";s*k+1
150 print "Al mattino ogni marinaio ne avra'";a
```

```
I marinai e la scimmia
N. marinai ?  3
Le noci sono 79
Al mattino ogni marinaio ne avra' 7
```

```
I marinai e la scimmia
N. marinai ?  2
Le noci sono 15
Al mattino ogni marinaio ne avra' 1
```

Super accuratezza

Normalmente, il vostro computer effettua calcoli con 6 o 7 cifre di accuratezza. Il calcolo in doppia precisione aumenta l'accuratezza fino a circa 13 cifre.
È tuttavia possibile effettuare i calcoli una cifra alla volta ed attribuire ogni cifra ad un elemento di un vettore. Si raggiungerà così virtualmente qualunque accuratezza si desideri, fino alla massima dimensione possibile per il vettore.
Questo programma esemplificativo effettua l'operazione piuttosto semplice di raddoppiare successivamente un numero (che è lo stesso che aggiungere 2 ad una potenza).
Sia 8192 il numero da rappresentare, allora:

A(4)	A(3)	A(2)	A(1)
8	1	9	2

Per aggiungere questo valore a se stesso, si sommano dapprima le cifre più a destra: A(1)+A(1). Se c'è un riporto, la variabile C è posta uguale ad 1, altrimenti a zero. Il totale è messo in B alla riga 100.
Se B è minore di 10, non c'è riporto (C=O) ed il nuovo A(1) è uguale a B. Se B è maggiore di 10 non c'è riporto (C=1) ed il nuovo A(1)=B—10.
Questa operazione è ripetuta per tutte le cifre (D) che compongono il numero e, al termine, il nuovo numero viene scritto alle righe 190-210.
Se A(N) fosse scritto seguito da un punto e virgola (;) la routine di stampa del Basic lascerebbe uno spazio prima di ogni cifra (per il segno) ed uno dopo (per leggibilità).
Poiché nel programma questi spazi non sono desiderati, viene usata la routine di stampa della riga 200, che scrive il valore convertito in stringa del valore ASCII di ogni cifra (che è poi corrispondente alla cifra stessa), ma senza spazi.
Incidentalmente, questo programma risolve la sfida del calcolo del numero di mosse nel problema delle Torri di Brahma (vedi "Cambio per ogni importo fino a $ 5.00").
Lo stesso approccio è anche usato nella prossima sezione, "Palindromi".

```
5 print chr$(14):rem setta minuscole
10 print "calcola 2 elevato alla N                    in superaccuratezza."
20 dim a(100)
30 m=0:c=0
40 d=0
50 for i=1 to 50:a(i)=0:next i
60 a(1)=1
70 i=0:c=0
80 m=m+1
90 i=i+1
100 b=a(i)+a(i)+c
110 if b<10 then c=0:goto 140
120 b=b-10
130 c=1
140 a(i)=b
150 if i<d then 90
160 if c=1 then 90
170 print m;
180 d=i
190 for n=i to 1 step -1
200 print chr$(a(n)+48);
210 next n
220 print
230 goto 70
```

```
calcola 2 elevato alla N in superaccuratezza.

1 2
2 4
3 8
4 16
5 32
6 64
7 128
8 256
9 512
10 1024
11 2048
12 4096
13 8192
14 16384
15 32768
16 65536
17 131072
18 262144
19 524288
20 1048576
21 2097152
22 4194304
23 8388608
24 16777216
25 33554432
26 67108864
27 134217728
28 268435456
29 536870912
30 1073741824
31 2147483648
32 4294967296
33 8589934592
34 17179869184
35 34359738368
36 68719476736
37 137438953472
38 274877906944
39 549755813888
40 1099511627776
41 2199023255552
42 4398046511104
43 8796093022208
44 17592186044416
45 35184372088832
```

Palindromi

Un palindromo è una parola, un verso o un numero che si legge allo stesso modo sia da sinistra a destra che viceversa. Per esempio, le parole "mom" e "eye" sono palindromi, come anche ogni riga del seguente verso:

Egad, a base life defiles a bad age
Doom an evil deed, liven a mood
Harass sensuousness, Sarah
Golf; No, sir, prefer prison-flog
Ban campus motto, "Bottoms up, MacNab" (*)

I palindromi numerici sono quei numeri che si leggono allo stesso modo sia in avanti che all'indietro. L'esame di questi numeri è un campo ricco di possibilità per l'utilizzo creativo del computer.
Un'ipotesi riguardante i palindromi solleva un'interessante domanda ancora senza risposta. Incominciate con un qualunque numero positivo intero. Se non è palindromo invertite le sue cifre e sommate i due numeri. Se la somma non è un palindromo, effettuate nuovamente la stessa operazione e continuate, finché non si raggiunga un palindromo. Per esempio, incominciando con 78:

```
   78
 + 87
 ----
  165
 +561
 ----
  726
 + 627
 ----
 1353
+3531
 ----
 4884
```

L'ipotesi, spesso assunta per vera, è che questo ciclo porterà sempre ad un palindromo. Ed infatti è proprio ciò che solitamente accade. La maggior parte dei numeri inferiori a 10.000 daranno un palindromo in meno di 24 addizioni. Ma c'è una spina nel fianco di questa ipotesi, il numero 196. Siete in grado di determinare se si potrà mai ottenere un palindromo partendo da 196?
Il numero 196 produrrà 1675 dopo due inversioni, ma dopo 100 inversioni la somma risultante ha 47 cifre e ancora non è palindroma. Perché citiamo 1675? Perché altri 10 numeri inferiori a 1000 conducono al valore 1675 e così non possono diventare palindromi. I primi cinque tra questi numeri sono 196, 295, 394, 493 e 592. Quali sono gli altri cinque?

Il presente programma accetta qualunque numero come valore iniziale e completa il ciclo consistente nell'aggiungere le successive inversioni e verificare se la somma è palindroma. Provate con qualche numero e vedete se riuscite ad identificare qualche caratteristica generale.

```
5 print chr$(14):rem setta minuscole
10 print"Verifica l'ipotesi secondo la quale la"
11 print"somma di numeri con le cifre invertite"
15 print "finisce col dare un Palindromo"
20 print
30 dim b(50)
40 print
50 input "Numero    ";a
60 e=0
70 e=e+1
80 a=a/10
90 if int(a)>0 then 70
100 for c=e to 1 step -1
110 a=a*10
120 b(c)=int(a-10*int(a/10))
130 next c
140 d=0
150 for c=1 to int(e/2)
160 if b(c)=b(e+1-c) then 180
170 d=1
180 next c
190 for c=e to 1 step -1
200 print chr$(b(c)+48);
210 next c
220 if d=1 then 260
230 print " Polidromo! "
250 goto 40
260 print " Non ancora"
270 if e/2<=int(e/2) then 290
280 b(int(e/2)+1)=2*b(int(e/2)+1)
290 for c=1 to int(e/2)
300 b(c)=b(c)+b(e+1-c)
310 next c
320 for c=1 to int(e/2)
330 b(e+1-c)=b(c)
340 next c
350 b(e+1)=0
360 for c=1 to e
370 b(c+1)=b(c+1)+int(b(c)/10)
380 b(c)=b(c)-10*int(b(c)/10)
390 next c
400 if b(e+1)<=0 then 140
410 e=e+1
420 goto 140
```

(*) Questo verso è abbastanza intraducibile in italiano. Più che un senso logico, in questo verso va visto un singolare esempio di palindromo. (N.d.T.).

```
Verifica l'ipotesi secondo la quale la
somma di numeri con le cifre invertite
finisce col dare un Palindromo


Numero     ?  34
34 Non ancora
77 Polidromo!

Numero     ?  161
161 Polidromo!

Numero     ?  360
360 Non ancora
423 Non ancora
747 Polidromo!

Numero     ?  120
120 Non ancora
141 Polidromo!

Numero     ?  74
74 Non ancora
121 Polidromo!

Numero     ?  135
135 Non ancora
666 Polidromo!

Numero     ?  185
185 Non ancora
766 Non ancora
1433 Non ancora
4774 Polidromo!

Numero     ?  18
18 Non ancora
99 Polidromo!

Numero     ?  30
30 Non ancora
33 Polidromo!

Numero     ?  365
365 Non ancora
928 Non ancora
1757 Non ancora
9328 Non ancora
17567 Non ancora
94138 Non ancora
177287 Non ancora
960058 Non ancora
1810127 Non ancora
9020308 Non ancora
17050517 Non ancora
88555588 Polidromo!
```

Usando questo metodo, scrivete un programma che esamini tutti gli interi compresi tra 1 e 10.000, escludendo quelli che in un qualunque momento diano 1675 per somma. Che cosa dimostra ciò? Tra l'altro, dovrete trovare un modo per trattare gli interi con 14 cifre, che sono più grandi di quanto il vostro computer può normalmente trattare.

4
Convergenza e ricorsività

Il computer è particolarmente adatto per effettuare calcoli ripetitivi e noiosi. Due approcci matematici per risolvere problemi riguardanti calcoli ripetitivi sono la convergenza e la ricorsività.
Alcuni problemi possono essere espressi piuttosto facilmente a parole o descritti con alcune semplici equazioni, ma con molte soluzioni possibili. Per esempio, in quanti modi potete cambiare una moneta americana da 10 cents? Questo problema è espresso in maniera semplice ed il numero di modi può essere elencato abbastanza facilmente: due nickels, oppure un nickel e cinque pennies, oppure 10 pennies, tre modi in tutto. Ma se volete trovare tutti i modi per cambiare un dollaro o cinque dollari, sarebbe bene avere un aiuto.
L'aiuto per questo genere di problemi viene da una classe di programmi che semplicemente spacca il problema in altri più piccoli e conta tutte le soluzioni alternative a seconda dell'insieme di regole. Ma una tecnica ancor più potente è quella ricorsiva. Con questa tecnica, viene definito un semplice algoritmo capace di risolvere un minimo sottoinsieme del problema. La potenza esclusiva di una routine ricorsiva deriva dalla capacità della routine di poter richiamare se stessa. Ciò è descritto più a fondo nel secondo programma di questa sezione.
Un altro approccio utile per risolvere problemi che non hanno una risposta esatta è quello delle approssimazioni successive. Per esempio, il valore esatto di π, e, o la lunghezza di una curva irregolare non possono essere determinati esattamente. Ma, per mezzo di approssimazioni sempre più accurate, è possibile avvicinarsi al valore desiderato dall'alto o dal basso o convergere da due direzioni. Gli ultimi quattro programmi in questo capitolo illustrano successive approssimazioni e convergenze.

Il cambio di un dollaro

Benché oggi si possa comprare ben poco con un penny (un millesimo di dollaro) questa moneta sarà ancora in circolazione per qualche tempo, essendo necessaria per arrotondare importi di tasse sulle vendite e per completare le collezioni numismatiche.
Oggi vengono coniate cinque monete negli U.S.A.: penny, nickel, dime, quarter e half dollar (rispettivamente pari ad un millesimo di dollaro, cinque cents, dieci cents, un quarto di dollaro e mezzo dollaro. N.d.T.). In quanti modi queste monete possono essere usate per cambiare un dollaro? Per esempio, un modo è due mezzi dollari, un altro è mezzo dollaro e due quarti, e così via. Fate un'altra ipotesi e scrivetela prima di leggere oltre.

Ci sono vari modi differenti per affrontare questo genere di problema. Uno consiste nel suddividerlo in problemi più piccoli e più facilmente risolvibili. In altre parole, in quanti modi si può cambiare un quarter? E un dime? Potete risolvere questi sottoproblemi e combinare le risposte così da ottenere la soluzione complessiva.
Se siete più portati per la matematica, potete scrivere una serie di equazioni ognuna relativa al cambio di ogni moneta rispetto ad ogni altra ed al dollaro e poi risolverle.
Un terzo approccio consiste nello scrivere combinazioni finché tutte le differenti combinazioni siano esaurite (o finché non vi esaurite voi stessi) e poi sommarle tutte.
Questo metodo potrebbe essere chiamato soluzione per esaurimento, ed è molto adatto per essere impostato su un computer.

Scrivete un programma che utilizza questo approccio per risolvere il problema. Se usate dei cicli, ad incrementi di uno, il computer potrebbe impiegare molto tempo per verificare tutte le combinazioni possibili, anche diverse ore.
Inoltre, se volete stampare tutte le combinazioni possibili, fate attenzione che anche la stampa potrebbe impiegare molto tempo ed una grande quantità di carta. Le combinazioni sono molte di più di quanto pensiate!
In effetti, la maggior parte della gente non sa indovinare la risposta a questo problema, e nemmeno sa avvicinarsi. Provate a chiedere ai vostri amici in quanti modi pensano sia possibile cambiare un dollaro. Registrate tutte le risposte e poi incolonnatele sul vostro computer. Qual è la media di tutte le risposte? E quali sono i valori estremi?

Il programma qui riportato utilizza il primo metodo discusso per risolvere il problema, in particolare suddividendo il problema in sottoproblemi e poi combinando le soluzioni in un'unica risposta finale.
Dapprima, il problema principale è suddiviso nel successivo problema minore riguardante il cambio in mezzi dollari. Ci sono tre soluzioni: niente mezzi dollari (H=O), un mezzo dollaro (H=1) e due mezzi dollari (H=2). L'ultima è banale essendoci un unico modo per risolverla, ma le altre due devono essere ulteriormente suddivise.

Ciò si ottiene dividendo il denaro rimanente in quarters e considerando i sottoproblemi successivi fino al livello più basso. All'aumentare del numero di sottoproblemi, ognuno di essi diviene più facile da risolvere. Infatti si raggiungono infine i sottoobiettivi, che possono essere risolti in un unico modo. Per esempio, se H=1, Q=1, D=2, e N=O, allora i pennies (P) devono essere cinque perché il totale sia 100.

Osservate che ai livelli di quarter, dime e nickel, vengono fatti aggiustamenti nei limiti dei cicli a seconda di quanto denaro rimane da cambiare. Per esempio, se H=1, gli unici possibili sottoobiettivi per i quarters sono 0, 1, e 2, ma non 3 o 4. Osservate anche che non è necessario verificare le combinazioni di monete per vedere se arrivano a totalizzare 100, né è necessario includere il penny come variabile. È sufficiente contare semplicemente il numero di sottoobiettivi, poiché ognuno di essi può essere risolto in un solo modo.

```
5 print chr$(14):rem setta minuscole
10 c=0
20 for h=0 to 2
30 for q=0 to 4-2*h
40 for d=0 to 10-5*h-2.5*q
50 for n=0 to 20-10*h-5*q-2*d
60 c=c+1
70 next n
80 next d
90 next q
100 next h
110 print "Un dollaro puo' essere cambiato in";c:print"modi differenti."
```

```
Un dollaro puo' essere cambiato in 292
modi differenti.
```

Provate ad effettuare qualche modifica a questo programma o scrivetene uno nuovo per risolvere i seguenti problemi. Supponiamo che vogliate due quarters nel vostro cambio per giocare a dei videogames. In quanti modi un dollaro può essere cambiato per ottenere almeno due quarters?

Visitando una piccola città, scoprite che i parchimetri accettano ancora i pennies. In quanti modi potete cambiare così da ottenere almeno tre pennies? È questo numero diverso da quello per ottenere cinque pennies? Supponete di voler fare anche una telefonata; in quanti modi potete cambiare un dollaro per ottenere almeno quattro pennies ed un dime?

Cambio di un importo fino a $ 5.00

Un altro modo per affrontare il problema del cambio presentato nella precedente sezione è per mezzo della tecnica di programmazione detta ricorsiva. Acquisite familiarità con questa tecnica — è molto potente! Donald Piele e Larry Wood descrissero questo metodo in un numero di *Creative Computing.*

Dapprima, definite le variabili che rappresentano la quantità di modi per cambiare n cents usando le monete indicate:

A = solo pennies
B = nickels e pennies
C = dimes, nickels e pennies
D = quarters, dimes, nickels e pennies
E = mezzi dollari, quarters, dimes, nickels e pennies

Inizialmente, vi sono due sottoproblemi nel fare il cambio per n cents. Nel primo, non vengono usati mezzi dollari, e D è il numero di modi per cambiare n cents. Nel secondo, quando vengono usati uno o più mezzi dollari, dopo che uno di essi è pagato, rimangono n-50 cents che possono essere pagati in E_{n-50} modi.

Poiché questi due casi si escludono a vicenda, si può inferire che $E_n=D_n+E_{n-50}$. Allo stesso modo:

$D_n=C_n+D_{n-25}$
$C_n=B_n+C_{n-10}$
$B_n=A_n+B_{n-5}$

Incominciate ora con il caso più semplice, arrivando a E_{100}. Innanzitutto, è facile comprendere perchè $E_0=1$. Quando n=50, $E_{50}=D_{50}+E_0$ ed è possibile cambiare 50 cents in un unico modo se i mezzi dollari sono consentiti. Quindi $E_0=1$. $D_0=C_0=B_0=A_0=1$. È anche vero che $A_n=1$ per tutti i valori di n in quanto c'è un unico modo per fare il cambio usando solo pennies. La relazione ricorsiva può essere ora usata per risolvere il problema originale.

Questa è la strategia usata nel progamma, che ha anche il vantaggio aggiuntivo di poter contare il numero di modi per cambiare (in monete) un qualunque importo.

Ed ora è il vostro turno. Riuscite a modificare il programma in modo da includere anche banconote, così da poter effettuare la stessa elaborazione per importi fino a $ 10.00?

Usando il metodo che preferite, scrivete un programma che effettui il cambio con un rublo. Le monete russe sono rispettivamente di 1,2,3,5,10,15,20,50 e 100 copechi. Vi sono 100 copechi in un rublo.

```
5 print chr$(14):rem setta minuscole
10 dima(101),b(101),c(101),d(101),e(101)
15 print
20 input"Valuta da cambiare ";x
30 m=int(20*x)+1
40 a(1)=1:b(1)=1:c(1)=1:d(1)=1:e(1)=1
50 for j=2 to m
60 a(j)=1
70 b(j)=a(j)+b(j-1)
80 c(j)=b(j)
90 if j<=2 then 110
100 c(j)=b(j)+c(j-2)
110 d(j)=c(j)
120 if j<=5 then 140
130 d(j)=c(j)+d(j-5)
140 e(j)=d(j)
150 if j<=10 then 170
160 e(j)=d(j)+e(j-10)
170 next j
180 print "Puoi cambiarlo in";e(m);"modi."
190 goto 15
```

```
Valuta da cambiare ?  2.23
Puoi cambiarlo in 3335 modi.

Valuta da cambiare ?  1.23
Puoi cambiarlo in 494 modi.

Valuta da cambiare ?  1.23
Puoi cambiarlo in 494 modi.

Valuta da cambiare ?  .23
Puoi cambiarlo in 9 modi.

Valuta da cambiare ?  1.23
Puoi cambiarlo in 494 modi.

Valuta da cambiare ?
```

Forse il problema più famoso utilizzato per dimostrare i principi della ricursione è quello delle Torri di Brahma, o Torri di Hanoi. Il problema è qui esposto nella forma possibilmente più simile all'originale. Dovreste essere in grado di risolverlo con un programma relativamente breve, usando la tecnica ricorsiva.
Nel grande tempio di Benares, sotto la cupola che indica il centro del mondo è posta una placca di ottone in cui sono fissati tre aghi di diamante, ognuno alto un cubito e grande come un'ape. Su uno di questi aghi, all'atto della Creazione, Dio pose 64 dischi di oro puro, dal diametro decrescente, con il più grosso sulla base della placca e gli altri sopra di esso. Questa è la Torre di Brahma.
Giorno e notte, incessantemente, i sacerdoti spostano i dischi da un ago all'altro, rispettando le immutabili leggi di Brahma. Queste leggi prescrivono che il sacerdote incaricato possa muovere un solo disco alla volta e che il disco spostato debba essere posto su un ago in modo tale che al di sotto non vi siano dischi di diametro inferiore. Quando i 64 dischi saranno così stati trasferiti dall'ago — su cui Dio li pose all'atto

della Creazione — su uno degli altri aghi, allora la torre, il tempio e i seguaci di Brahma si sgretoleranno polverizzandosi e, con un grande tuono, il mondo si disintegrerà.
Supponendo che i sacerdoti effettuino uno spostamento ogni secondo, lavorando 24 ore al giorno tutti i giorni dell'anno, impiegherebbero 58.454.204.609 decenni ed oltre 6 anni per completare il loro compito, ipotizzando che non commettano mai errori, in quanto una sola piccola svista comprometterebbe tutto il lavoro.
Quanti spostamenti sono necessari perché si avveri la profezia? Provate il vostro programma con meno di 64 dischi per accertarvi di essere sulla strada giusta. Ecco una tabella riportante i primi spostamenti:

Dischi	*Mosse*	*Dischi*	*Mosse*
1	1	6	63
2	3	7	127
3	7	8	255
4	15	9	511
5	31	10	1023

Convergenza ad "e" e π

Una costante matematica di incredibile importanza è rappresentata dalla piccola lettera "*e*". Questa costante è sia irrazionale che trascendentale. Se volete, cercate questi termini in un dizionario o un testo di matematica, oppure arrischiatevi direttamente nel prossimo paragrafo.
La costante "e" fu inizialmente definita da John Napier, l'inventore dei logaritmi, al quale dobbiamo una eterna gratitudine. Perché? Se "e" non fosse stata scoperta, i progressi in matematica, fisica ed astronomia sarebbero rimasti indietro di un secolo o più, perché "e" è la base di tutti i logaritmi naturali e questi logaritmi sono la base di molte branche della scienza e della matematica.
Come si calcola "e"? La costante "e" è il limite della seguente espressione, per n tendente all'infinito.

$$e = (1+1/n)^n$$

Il suo esatto valore non potrà mai essere trovato ma, calcolato con 15 cifre decimali, risulta pari a 2,718281828459045... Come può essere calcolata "e"? Incominciate con 1+1/2 elevato al quadrato, ottenendo 2+1/4. Poi elevate al cubo 1+1/3 e ottenete 2.3686. Elevando 1+1/4 alla quarta potenza si ottiene 2.414, e così via. Scrivete un programma per questo metodo e fategli stampare il valore iniziale di "e" ed ogni valore derivante da ogni successiva frazione sommata.
Un altro approccio consiste nell'espandere l'espressione usando il teorema binomiale, e di nuovo, facendo tendere n all'infinito. L'espressione per l'espansione è:

$$e = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} \cdots \frac{1}{n!}$$

Ecco nuovamente dove il computer può essere di aiuto. Poiché 3! è 3*2! e 4! è 4*3!, non è necessario rifare tutti i calcoli per ogni successiva frazione. Osservate il programma facendo particolarmente attenzione ai calcoli contenuti nelle istruzioni 70 e 90.

```
5 print chr$(14):rem setta minuscole
10 print "Convergenza ad e"
20 e=1
30 i=0
40 i=i+1
50 d=1
60 for j=1 to i
70 d=d*j
80 next j
90 e=e+1/d
100 print e
110 goto 40
```

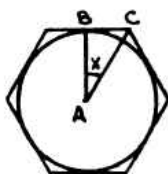
```
Convergenza ad e
 2
 2.5
 2.66666667
 2.70833334
 2.71666667
 2.71805556
 2.71825397
 2.71827877
 2.71828153
 2.7182818
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
 2.71828183
```

Pi greco (π) è un'altra importante costante matematica irrazionale (che significa che il suo esatto valore non può essere determinato) e trascendente (cioè non è la soluzione di alcuna equazione algebrica). È interessante osservare che π era conosciuto ed usato già dagli antichi. Archimede, che visse nel secondo secolo avanti Cristo, dimostrò — con un poligono regolare di 96 lati (quasi un cerchio) — che il valore di π era inferiore a 22/7 e superiore a 3 — 10/71, risultato notevole per la matematica del suo tempo.

Nel 150 d.C. Tolomeo utilizzò il valore di 3.1416 per π e verso la metà del sedicesimo secolo fu scoperta la frazione 355/113, con un risultante valore di π accurato fino alla sesta cifra decimale.
A titolo di curiosità ricordiamo che nel 1897 l'Assemblea Generale dello stato americano dell'Indiana stabilì con un decreto che il valore di π era 4.
Varie serie infinite possono possono essere usate per ottenere valori sempre più accurati di π. Una di queste serie è (1—1/3 + 1/5—1/7 + 1/9...). Questa è definita una successione aritmetica, che converge molto lentamente. Il programma qui riportato mostra solo un valore ogni 500 della successione. Non allarmatevi se il programma non sembra girare molto velocemente; una notevole quantità di calcoli viene infatti svolta tra ogni valore mostrato.

```
5 print chr$(14):rem setta minuscole
10 print chr$(147)
20 print "Convergenza a PI greco per serie          aritmetica";
30 s=1
40 i=1
50 q=0
60 p=0
70 q=q+1
80 p=p+s/i
90 i=i+2
100 s=-s
110 if q<499 then 70
120 q=0
130 print:print p*4;
140 goto 70
```

```
Convergenza a PI greco per serie          aritmetica
 3.14359667
 3.14059066
 3.14226067
 3.14109167
 3.14199346
 3.14125867
 3.14187896
 3.14134218
 3.14181534
 3.14139227
 3.14177485
 3.141?2568
 3.141746[illegible]
 3.14144 55[illegible]
```

In realtà, l'approccio usato da Archimede converge molto più rapidamente e, con l'aiuto di un computer, è possibile andar ben al di là del poligono a 96 lati usato da Archimede.

Il suo approccio consisteva nel costruire poligoni inscritti e circoscritti e misurarne i perimetri per approssimare la circonferenza di un cerchio. Considerate un poligono circorscritto ad un cerchio di raggio 1. Il perimetro è pari alla lunghezza di un lato moltiplicata per il numero dei lati. Essendo la tangente di x = AB/BC, ma essendo BC=1, allora tan(x)=AB e la lunghezza di un lato è pari a 2*tan(x). Poiché la circonferenza è uguale a $2\pi r$ e r=1, allora π è la circonferenza (o il perimetro di un poligono di n lati) diviso per 2.

La trigonometria ci porta a concludere che il perimetro di un poligono inscritto è pari al numero di lati per sin(x)xcos(x).
Il secondo programma produce valori di π usando poligoni inscritti e circoscritti. Sfortunatamente, c'è un grosso punto debole nel programma, perché i gradi vanno convertiti in radianti nell'istruzione 40. Ciò significa, naturalmente, che dovete già conoscere il valore di π, in quanto il fattore di conversione è 360° diviso per 2 π.
Lasciando da parte questo difetto, è interessante notare come questo programma converga più rapidamente al valore di π, rispetto al precedente. Questo perché la convergenza avviene geometricamente anziché aritmeticamente.
Riuscite ad immaginare una convergenza geometrica al valore di π, che non richieda che voi già conosciate il valore di π stesso (o un fattore di conversione) prima di iniziare?

```
5 print chr$(14):rem setta minuscole
10 print "Convergenza a PI greco per poligoni.":print
15 print " Inscritti","circoscritti":print
20 n=6
30 n=2*n
40 x=360/(n*57.29578)
50 print n*sin(x)*cos(x)/2,
60 print n*tan(x)/2
70 goto 30
```

```
Convergenza a PI greco per poligoni.

 Inscritti              circoscritti

 2.5980762               3.46410158
 2.99999998               3.21539028
 3.10582851               3.15965992
 3.13262859               3.14608619
 3.13935018               3.14271458
 3.14103193               3.14187303
 3.14145245               3.14166272
 3.14155758               3.14161015
 3.14158386               3.141597
 3.14159043               3.14159372
 3.14159208               3.1415929
 3.14159247               3.14159268
 3.14159257               3.14159263
 3.1415926               3.14159261
 3.14159246               3.14159247
 3.14159247               3.14159247
 3.1415919               3.14159191
```

```
 3.1415919               3.14159191
 3.14158966               3.14158966
 3.14158966               3.14158966
 3.14158067               3.14158067
 3.14158067               3.14158067
 3.14154459               3.14154484
 3.14154465               3.14154478
 3.14140088
```

Una revisione della convergenza a π

In risposta alla domanda posta nell'ultimo paragrafo del capitolo precedente ecco un metodo per convergere a π senza conoscerne in anticipo il valore.
Come nel programma precedente, l'approccio di base consiste nel sommare la lunghezza dei lati di un poligono inscritto e dividere per 2r, ottenendo così il valore di π. Il programma inizia con un quadrato (quattro lati) e raddoppia ogni volta il numero di lati. Se la lunghezza del lato è S, allora la lunghezza S di un lato del poligono successivo con il doppio di lati si ottiene applicando il teorema di Pitagora.
In particolare:

$$X^2 + (S/2)^2 = R^2$$

$$(R-X)^2 + (S/2)^2 = (S')^2$$

così,

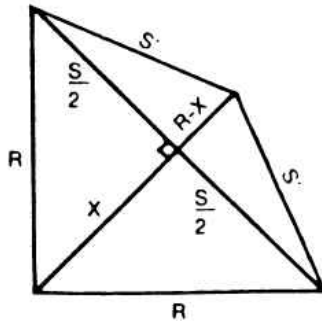
$$S' = \sqrt{(R-\sqrt{R^2 - (S/2)^2})^2 + (S/2)^2}$$

È facile ridurre algebricamente questa formula, ma così si riduce l'accuratezza. Inoltre, potrete vedere che S*S è calcolato con accuratezza leggermente superiore a S^2.
Sfortunatamente, la ricerca della massima accuratezza è piuttosto controversa su un computer che non dispone di aritmetica in doppia precisione. Osservate anche che l'accuratezza non aumenta con più di 1024 lati e, infatti, quando i valori nei calcoli eccedono la capacità del computer (4194304 lati) l'accuratezza incomincia a diminuire notevolmente.
C'è ancora un altro metodo per calcolare π per convergenza, utilizzando le scoperte di Gregorio ed Eulero. Gregorio scoprì la formula dell'arctangente:

$$\text{Arctan}(x)=x—x^3/3+x^5/5 +x^7/7+...$$

Eulero giunse ad una formula piuttosto interessante per π:

$$\pi=4(\arctan(1/2)+\arctan (1/3))$$

Provate a combinare queste due formule per calcolare π. Se siete molto abili potete ottenere un'accuratezza molto superiore alle sette cifre decimali ottenute dai programmi fin qui presentati.

```
5 print chr$(14):rem setta minuscole
10 print "Convergenza a PI greco                    per poligoni inscritti."
20 print " Lati"," Perimetro":print
30 r=10
40 z=2
50 n=z+z
60 s=r*sqr(z)
70 for k=1 to 22
80 print n,(s*n)/(z*r)
90 y=s*s/(z*z)
100 x=r-sqr(r*r-y)
110 s=sqr(x*x+y)
120 n=n*z
130 next k
```

```
Convergenza a PI greco                    per poligoni inscritti.

 Lati             Perimetro

 4              2.82842713
 8              3.06146746
 16              3.12144515
 32              3.13654849
 64              3.14033116
 128              3.14127725
 256              3.1415138
 512              3.14157294
 1024              3.14158773
 2048              3.14159142
 4096              3.14159235
 8192              3.14159258
 16384              3.14159264
 32768              3.14159265
 65536              3.14159265
 131072              3.14159265
 262144              3.14159265
 524288              3.14159265
 1048576              3.14159265
 2097152              3.14159265
 4194304              3.14159265
 8388608              3.14159265
```

Lunghezza di una curva

I programmi precedenti hanno dimostrato come sia possibile calcolare un valore molto accurato di π sommando la lunghezza dei lati di un poligono, al suo approssimarsi ad un cerchio e dividendo per 2r.
Usando un approccio simile, dovrebbe essere possibile inscrivere un poligono, o parti di un poligono, in ogni curva regolare, determinando così la lunghezza della curva.
Questo programma determina per approssimazione la lunghezza di qualunque curva, come definita nell'istruzione 100, dividendola in un numero crescente di sottointervalli e calcolando la somma delle secanti (una linea retta che tagli una curva in due o più punti).
Per eseguire il programma dovete inserire la formula o l'equazione che descrive la vostra curva nella linea 100 nella forma:

100 DEF FNA (X) = la vostra funzione di X

Poi battete RUN ed inserite gli estremi della curva che volete usare nei vostri calcoli.
Questi punti estremi sono inseriti sotto forma di ascissa, cioè la coordinata orizzontale (x) del punto.
Il programma è scritto in modo da sommare le successive lunghezze delle secanti e calcolare la percentuale di incremento in ogni somma a confronto con la precedente.
Nell'esempio si usa la funzione $2x^3+3x^2-2x+3$. Osservate come la lunghezza del calcolo aumenti notevolmente all'aumentare del numero di intervalli da 2 a 16, mentre aumenta di poco oltre il 16.
Provate questo programma con diverse curve e funzioni. Potrebbe essere d'aiuto disegnare prima la funzione (ricordate il programma per farlo?) e poi calcolarne la lunghezza. Questo metodo è più accurato per una funzione che non cambia nella direzione della curva all'interno dell'intervallo scelto oppure per una funzione con una o più variazioni? Perché?

```
5 print chr$(14):rem setta minuscole
10 print chr$(147)
20 print "Lunghezza della curva "
30 print "la cui funzione e' definita                alla linea 100"
40 print
50 print
60 input "Ascisse ai valori estremi";p,q
70 print
80 print " Inter-","Lunghezza","  Cambio %"
85 print " valli"," secante","in lunghezza"
90 print
95 s1=0
100 def fna(x)=2*x↑3+3*x↑2-2*x+3
110 for n=1 to 9
120 e=2↑(n-1)
130 h=(q-p)/e
140 s=0
150 for i=0 to e-1
160 l=sqr((fna(p+i*h+h)-fna(p+i*h))↑2+h*h)
170 s=s+l
180 next i
190 if s1>0 then220
200 print e;tab(8);s;tab(18);"nessun valore"
210 goto 240
220 p5=((abs(s1-s))/s1)*100
230 print e;tab(8);s;tab(18);p5
240 s1=s
250 next n
```

```
Lunghezza della curva
la cui funzione e' definita                alla linea 100


Ascisse ai valori estremi ? -1,6

 Inter-          Lunghezza           Cambio %
 valli           secante          in lunghezza

 1          525.046666              nessun valore
 2          525.158263               .0212546484
 4          529 652248               .855739211
 8          531.017134               .257694752
 16          531.964256               .178360033
 32          532.016566               9.83336555e-03
 64          532.041679               4.72022717e-03
 128          532.048562               1.29376759e-03
 256          532.050272               3.21342778e-04
```

La convergenza applicata al calcolo della radice quadrata

Poiché la maggior parte delle radici quadrate sono irrazionali, i metodi usati per calcolarle comportano di solito successive approssimazioni. Potete ovviamente richiamare semplicemente la funzione di radice quadrata in Basic o su molti calcolatori tascabili, ma è interessante verificare vari metodi per calcolare le radici quadrate senza queste funzioni già pronte. Dopo tutto, queste funzioni non sono altro che routine di approssimazioni successive già installate nella macchina.
Ovviamente, la radice quadrata è l'operazione inversa del quadrato di un numero. Tutti i metodi di calcolo di radici quadrate utilizzano questo fatto, ma il modo in cui esso è impiegato è piuttosto differente nei vari computer e calcolatrici.
Il programma calcola un limite inferiore e superiore per la radice quadrata di un numero e, per successive approssimazioni, restringe il valore della radice entro intervalli sempre più piccoli finché raggiunge il livello desiderato di accuratezza.
Il valore inziale per il limite inferiore è 0 e per il limite superiore è il numero Z di cui si ricerca la radice quadrata. Il programma poi divide questo intervallo in dieci parti, semplicemente dividendo per 10 la differenza tra i due numeri. La variabile I viene incrementata dal valore inferiore a quello superiore con incrementi S. Se, in un punto qualunque, il quadrato di I supera Z, viene posto un nuovo limite superiore pari ad I ed un nuovo limite inferiore pari a I-S.

Questo metodo converge molto rapidamente ed aggiunge approssimativamente una cifra decimale di accuratezza ad ogni passo dopo il terzo. Che succede se inserite nel programma un numero con radice quadrata esatta, come 25 o 49? Perché?
Un altro approccio per il calcolo delle radici quadrate per successive approssimazioni è di partire con una radice di prova, X. Se X*X è inferiore al numero originale N, allora incrementa il valore di prova di 0.1. Se X*X è maggiore di N, allora ritorna al valore precedente. Questa è la prima cifra della radice. Ora incominciate ad avanzare di 0.01. Continuando in questo modo, ogni volta si individua una cifra, finché non si raggiunge la precisione desiderata.

Questo metodo è piuttosto adatto e veloce per radici di numeri inferiori ad 1. Un buon valore iniziale di prova è 0.1. Ma è adatto per numeri più alti? Come va determinato un valore iniziale di prova? In questo metodo, soprattutto per numeri maggiori di 10, il valore iniziale di prova per la radice ha una grande influenza nella determinazione del tempo che verrà impiegato perché il calcolo arrivi a convergere. Scrivete un programma operante in base a questo metodo e confrontatene la velocità e l'accuratezza con il programma descritto nel testo.

```
5 print chr$(14):rem setta minuscole
10 print "radici quadrate"
20 input "Il tuo numero";z
30 e=.00001
40 print "Limite basso","Limite alto"
50 a=0
60 b=z
70 s=(b-a)/10
80 print a;tab(18);b
90 if abs(a*b-z)<e then 180
100 for i=a to b step s
110 if z<i*i then 150
120 next i
130 b=b*10
140 goto 70
150 b=i
160 a=i-s
170 goto 70
180 print:print "accuratezza: .00001"
190 print "valore medio:";(a+b)/2
```

```
radici quadrate
Il tuo numero ? 54
Limite basso          Limite alto
 0                     54
 5.4                   10.8
 7.02000001            7.56000001
 7.344                 7.398
 7.344                 7.3494
 7.34832               7.34886
 7.348428              7.348482
 7.3484658             7.3484712
 7.34846904            7.34846958

accuratezza: .00001
valore medio: 7.34846931
```

5
Calcoli composti

Come per le approssimazioni successive e le ricursioni, anche il calcolo composto comporta molte elaborazioni ripetitive. Alcuni interessi composti e certe situazioni di crescita possono essere rappresentate da una formula, ma per molti problemi la soluzione con elaborazioni ripetitive è un approccio eccellente.
Non c'è niente di magico nel calcolo composto: si parte generalmente con un importo iniziale di denaro, quantità di animali, ecc. Questa quantità cresce poi o diminuisce di una certa percentuale ad intervalli prefissati.
Il nuovo importo è pari al vecchio importo aumentato o diminuito di questa percentuale. Si ripete continuamente questo calcolo, risolvendo così il problema.
In questo capitolo sono descritti cinque diversi tipi di problemi che comportano qualche tipo di calcolo composto. Provate i problemi posti in aggiunta ad alcuni programmi; rimarrete sorpresi dai risultati.

Gli indiani e gli interessi

Ecco un semplice problema di interesse composto che conduce ad un sorprendente risultato. Il problema riguarda la vendita dell'isola di Manhattan agli olandesi in cambio di cianfrusaglie e perline del valore di circa $24.
Se i $24 che gli Indiani ricevettero nel 1626 fossero stati depositati in banca al tasso annuo di interesse composto del 5.75%, a quanto ammonterebbe nel 1983?
Il programma qui riportato risolve questo piccolo problema facendo uso della formula per calcolare l'interesse composto. In particolare, se P dollari vengono investiti ad un tasso annuale R (espresso in forma decimale) e composti N volte, l'ammontare totale A è dato dalla formula:

$$A = P(1 + R)^N$$

Quanto si guadagnò nel solo 1983? Quanto fu il guadagno nel decennio dal 1974 al 1983? Potete modificare il valore di N nell'istruzione 30 per ottenere le risposte a queste domande. Sarebbe tuttavia meglio inserire l'anno finale con un'istruzione INPUT.
Sapete leggere un numero espresso nel formato esponenziale (E)? In un formato più convenzionale il numero è $11.176.500.000, ovvero 11,2 miliardi di dollari.
Potete migliorare il programma in molti modi, rendendolo più adatto a calcoli generici di interesse composto. Modificatelo in modo che possa accettare qualunque anno di inizio e fine calcolo, qualunque tasso di interesse e qualunque importo di capitale iniziale.
Così come è espresso, il problema è un po' irrealistico, perché le banche non hanno in realtà pagato lo stesso tasso del 5.75% dal 1626 al 1983. Modificate il programma in modo che calcoli l'importo totale basandosi sui seguenti tassi di interesse:

Anni	*Tasso di interesse*
1626-1830	1.5%
1831-1870	2.0%
1871-1910	3.0%
1911-1921	3.5%
1922-1929	6.5%
1930-1940	2.3%
1941-1945	3.5%
1946-1960	5.3%
1961-1980	6.5%
1981-1983	9.5%

```
5 print chr$(14)
10 p=24
20 r=.0575
30 n=1983-1626
40 a=p*((1+r)↑n)
50 print"gli indiani ricevettero $24 nel 1626     per NY."
60 print:print"nel 1983 al 5.75%                        sarebbero $";a
```

```
gli indiani ricevettero $24 nel 1626 per NY.

nel 1983 al 5.75% sarebbero $ 1.11764926e+10
```

Risparmi sistematici

Nel precedente programma l'interesse composto era calcolato per mezzo di una formula ben nota ai banchieri, a chi effettua prestiti ed agli agenti immobiliari. Tuttavia, se non aveste conosciuto la formula, come avreste calcolato, a mano o con una calcolatrice, gli interessi sul denaro depositato su un conto di risparmio?
Avreste probabilmente incominciato moltiplicando il capitale P per il tasso R e sommando quell'importo al capitale originario all'inizio del secondo anno.
Ripetendo questa operazione per tutti gli anni in cui il denaro è in banca si otterrà l'importo finale. Perché non scrivere un programma che effettui i calcoli in questo modo invece che usare una formula? Eccolo, questo programma.
È impostato per un capitale di 100 (istruzione 50), un tasso del 10% (R=0.1 nell'istruzione 60) e dieci anni (N=10 nell'istruzione 70). Naturalmente questi valori potrebbero essere letti usando istruzioni INPUT. L'intelligente calcolo della linea 100 arrotonda l'importo a due cifre decimali (dollari e cents).
A differenza del programma della sezione precedente, questo non fa uso di una formula ad interesse composto, ma semplicemente somma ogni anno l'interesse al capitale incrementato nella linea 90.

```
5 printchr$(14)
10 printchr$(147);"Calcolo interesse su $100 investiti al  10% per 10 anni."
30 print"Alla fine       Saldo"
40 print"dell'anno      attuale"
50 p=100
60 r=.1
70 n=10
80 fori=1ton
90 p=p+p*r
100 printi;tab(14)int(p*100+.5)/100
110 nexti
```

```
Calcolo interesse su $100 investiti al  10% per 10 anni.
Alla fine      Saldo
dell'anno     attuale
 1             110
 2             121
 3             133.1
 4             146.41
 5             161.05
 6             177.16
 7             194.87
 8             214.36
 9             235.79
 10             259.37
```

Ora, come si potrebbe modificare questo programma per ottenere un piano di risparmi sistematici? In altre parole, ogni anno aggiungete $100 al capitale iniziale. Con questo programma, è facile fare la modifica per i risparmi sistematici: basta aggiungere l'istruzione 105 per sommare il nuovo deposito ogni anno al capitale che progressivamente aumenta.

```
5 printchr$(14)
10 printchr$(147);"Calcolo interesse su $100 investiti al"
15 print"10% per 10 anni"
30 print"Alla fine    Saldo"
40 print"dell'anno   attuale"
50 p=100
55 c=100
60 r=.1
70 n=10
80 fori=1ton
90 p=p+p*r
100 printi;tab(4);i*c;tab(12);int(p*100+.5)/100
105 p=p+c
110 nexti
```

```
Calcolo interesse su $100 investiti al
10% per 10 anni
Alla fine    Saldo
dell'anno   attuale
 1    100      110
 2    200      231
 3    300      364.1
 4    400      510.51
 5    500      671.56
 6    600      848.72
 7    700      1043.59
 8    800      1257.95
 9    900      1493.74
 10    1000      1753.12
```

Nel tentativo di attirare clienti, molte banche hanno incominciato ad offrire, negli ultimi 20 anni, conti di risparmio e di investimento in cui gli interessi sono capitalizzati più spesso che annualmente. All'inizio degli anni '50 molte banche giunsero alla capitalizzazione trimestrale e, verso la fine degli anni '50, anche giornaliera; nei primi anni '60 alcuni istituti particolarmente competitivi giunsero alla capitalizzazione continua.

Che cosa comporta in realtà una capitalizzazione più frequente? Modificate uno dei due programmi così da calcolare l'interesse per capitalizzazioni più frequenti. Qual è la differenza di interessi per un anno con capitalizzazione rispettivamente annuale, trimestrale, mensile, giornaliera e continua (ogni secondo), su un capitale di $1000 investiti all'8%? E per dieci anni?

Incidentalmente, se volete usare il metodo della formula illustrato nella sezione precedente, la formula per il capitale P investito al tasso R capitalizzato N volte all'anno è:

$$A = P(1 + R/N)^N$$

Provate a risolvere il seguente problema con l'uso degli stessi principi. I prezzi al consumo aumentarono in media del 8.8% nel 1980. (*)Il governo cerca continuamente di tenere l'inflazione sotto controllo ma, sembra, senza molto successo. Ipotizzando che i prezzi continuino a salire allo stesso modo (8.8%) ogni anno, quanto costerà un'utilitaria da $6000 (in dollari 1980) nell'anno 2000? Quanto costerà quanto voi avrete 65 anni?

(*): questo dato è evidentemente relativo all'economia U.S.A. In Italia i tassi di inflazione sono, come noto, ben superiori! (N.d.T.).

Una revisione dei risparmi sistematici

Usando la formula per gli interessi composti ed i risparmi sistematici, questo programma calcolerà l'importo accumulato dopo un certo periodo di tempo.
Quando eseguite il programma, esso chiederà quanto volete risparmiare ogni mese, il numero di periodi di capitalizzazione in un anno, il tasso di interesse e per quanto tempo volete proseguire nel vostro programma di risparmi sistematici. Il programma poi calcolerà l'importo totale al termine di tale periodo.
Dai programmi precedenti, dovreste riuscire a comprendere come un programma di risparmi sistematici sia un modo molto efficace per accumulare un bel gruzzolo per il futuro.

```
5 printchr$(14)
10 printchr$(147);"Calcola gli interessi e il saldo per un"
20 input"Deposito mensile, $";a
40 input"Periodi di calcolo composto  ";b
80 input"Tasso di interesse (xx.x) ";r
100 r=r/100
110 input"N. anni ";n
130 print
140 print"Alla fine        Saldo"
150 print"dell'anno       attuale"
160 print
170 c=12*a
180 p=12*a
190 fori=1ton
200 p=p*(1+r/b)↑b
210 printi;tab(6);i*c;tab(14);int(p*100+.5)/100
220 p=p+c
230 nexti
```

```
Calcola gli interessi e il saldo per un
Deposito mensile, $?  10
Periodi di calcolo composto  ? 1
Tasso di interesse (xx.x) ? 8.5
N. anni ? 8

Alla fine        Saldo
dell'anno       attuale

 1   120      130.2
 2   240      271.47
 3   360      424.74
 4   480      591.04
 5   600      771.48
 6   720      967.26
 7   840      1179.68
 8   960      1410.15
```

```
Calcola gli interessi e il saldo per un
Deposito mensile, $?  10
Periodi di calcolo composto  ? 365
Tasso di interesse (xx.x) ? 8.5
N. anni ? 8

Alla fine         Saldo
dell'anno         attuale

 1    120       130.64
 2    240       272.88
 3    360       427.73
 4    480       596.32
 5    600       779.86
 6    720       979.68
 7    840       1197.23
 8    960       1434.08
```

Provate a combinare ciò che avete imparato dai programmi di queste sezioni per scrivere un programma completamente generalizzato per i risparmi sistematici. Dovrebbe richiedere in input i seguenti dati:

- Deposito iniziale.
- Frequenza dei depositi periodici.
- Ammontare di ogni deposito.
- Tasso di interesse.
- Frequenza di capitalizzazione degli interessi.
- Durata del programma di risparmi.

Il vostro programma dovrebbe produrre l'output in forma tabulare mostrando gli anni (o altri periodi di tempo), l'importo investito senza interessi, l'importo totale con interessi ed i soli interessi.

Pagamenti di prestiti

Risparmiare denaro in modo sistematico è un nobile obiettivo, tuttavia molte persone si ritrovano frequentemente a colloquio con un altro funzionario di banca, cioè quello che si occupa di prestiti.
Questo programma calcola la rata da pagare per un prestito relativo ad un periodo di un anno o più. Il programma vi chiede i punti chiave del prestito in questione: importo prestato, tasso annuale di interesse, intervallo di tempo tra una rata e l'altra, durata del prestito in anni.
L'esempio mostra un prestito a termine relativamente breve (2 anni) di $3000 ad un tasso concordato dell'8.5%. La rata mensile del prestito è calcolata in $136.37 e l'interesse totale in $272.79. Perché l'interesse totale non è $255 (l'8.5% di $3000)? È invece il 9.09%. Si tratta di un errore nel programma?
Provate il programma con qualche prestito di maggiore durata, a tassi di interesse reali. Per esempio, eseguitelo per un prestito per l'acquisto di un'automobile da $8000 al 12% per un periodo di 5 anni. Forse potete dedurne che risparmiare sistematicamente il vostro denaro e pagare in contanti ha più senso che non effettuare pagamenti rateali.
Eseguite il programma per calcolare la rata da pagare per l'ammortamento di una casa da $150.000 con un anticipo di $40.000. Provate con un tasso di interesse del 16% per un periodo di 30 anni. Ahimè! Guardate tutti quegli interessi!
Il nucleo del programma è costituito dalle istruzioni 180-210. Riuscite a capire che accade in questi calcoli?

```
5 printchr$(14)
10 print"Calcola il pagamento di un prestito."
20 print
30 input"Importo prestato ";a
40 input"Tasso di interesse (xx.x) ";i
50 input"Periodicita' (mesi)  ";p
60 input"Periodo (anni) ";y
70 input"Tabella (1) o solo totali(2) ";v
80 print
150 ifv=2then180
160 print"Periodo Interesse Capitale"
170 print"         dovuto    dovuto
180 z=(y*12)/p
190 k=(i*(p/12))/100
200 e=a*k/(1-1/(1+k)↑z)
210 e=int(e*100+.5)/100
220 c=a
230 f=0
240 d1=0
250 t1=0
260 t1=t1+1
270 ift1>zthen380
280 b=t1
290 c=c-f
300 d=c*k
310 f=e-d
320 d=int(d*100+.5)/100
330 f=int(f*100+.5)/100
340 d1=d1+d
350 ifv=2then260
360 printb;tab(8);d;tab(19);f
370 goto260
380 ifv=1then430
390 d1=int(d1*100+.5)/100
400 print"Capitale  $";a
410 print"Interessi $";d1
420 goto450
430 print"         --------- --------"
440 print"Totale   ";d1;tab(18);a:print
450 e5=int((d1+a)*100+.5)/100
460 e6=e5/((y*12)/p)
470 e6=int(100*e6+.5)/100
480 print"Totale  $";e5
490 print"Per pagamento  $";e6
```

```
Calcola il pagamento di un prestito.

Importo prestato ? 3000
Tasso di interesse (xx.x) ? 8.5
Periodicita' (mesi)  ?1
Periodo (anni) ? 2
Tabella (1) o solo totali(2) ?2

Capitale  $ 3000
Interessi $ 272.79
Totale  $ 3272.79
Per pagamento  $ 136.37
```

Interessi sulle vendite a credito

Troppo spesso il tasso di interesse addebitato su un prestito o una vendita a credito è nascosto in una nota scritta in piccolo. Dopo tutto la banca, il rivenditore d'auto o la compagnia finanziaria vogliono convincervi che siete in grado di affrontare la spesa relativa ad una nuova auto, casa o qualsivoglia altra cosa.
Questo programma calcola il tasso d'interesse su un prestito dato il capitale del prestito, il numero di rate e l'importo per rata. Per rendere le cose ancora più facili, il programma richiede il prezzo di acquisto in contanti dell'articolo e l'anticipo, e automaticamente calcola l'importo del prestito.
L'esempio mostra un prestito piuttosto irrealistico per un oggetto del valore di \$ 88.99. L'anticipo fu di \$10.00 ed il prestito è su un periodo di 18 mesi; ogni pagamento mensile è di \$4.85. L'esecuzione del programma mostra che il tasso di interesse reale è un modesto 7.01%.

Provate ora il programma con un prestito più realistico. Una compagnia che opera contro le termiti nel New Jersey pubblicizza un trattamento completo per la vostra casa a soli \$200; soltanto \$29.95 di anticipo e 24 rate mensili da \$11.95 ciascuna. È un offerta così stupefacente come sembra? Qual è il tasso annuale di interesse?

```
5 printchr$(14)
10 printchr$(147);"Tasso di interesse su vendite a credito"
20 print:input"Prezzo di acquisto";p
30 input"Anticipo";d
40 input"Numero di rate";n
50 input"Numero di rate al mese";m
60 input"Importo per rata";a
70 p1=p-d
80 t=a*n-p1
90 y=n/(m*12)
100 r=int(10000*t/p1/y+.5)/100
110 print:print"Il tasso di interesse e'";r;"%"
```

```
Tasso di interesse su vendite a credito

Prezzo di acquisto ? 1000
Anticipo ? 0
Numero di rate ? 24
Numero di rate al mese ? 2
Importo per rata ? 50

Il tasso di interesse e' 20 %
```

```
Tasso di interesse su vendite a credito

Prezzo di acquisto ? 88.99
Anticipo ? 10
Numero di rate ? 18
Numero di rate al mese ? 1
Importo per rata ? 4.85

Il tasso di interesse e' 7.01 %
```

La crescita della popolazione

Finora tutti i problemi di capitalizzazione in questo capitolo hanno riguardato il denaro — interessi, risparmi e prestiti. Ma vi sono molti affascinanti problemi analoghi che non riguardano il denaro. Considerate ad esempio il seguente problema. Nel 1960 il numero di abitanti degli Stati Uniti e del Messico erano rispettivamente di 180 e 85 milioni. Il tasso annuo di crescita della popolazione era dell'1.23% negli Stati Uniti e del 2.23% per il Messico. Se questi tassi di crescita rimanessero stabili, in qualche lontano anno la popolazione del Messico supererà quella degli Stati Uniti. In quale anno ciò accadrà?
Il programma usa il metodo di accumulare l'importo del capitale piuttosto che una formula, e lo applica ad entrambe le popolazioni. L'esempio rivela che la popolazione del Messico supererà quella degli U.S.A. in un futuro non troppo distante.

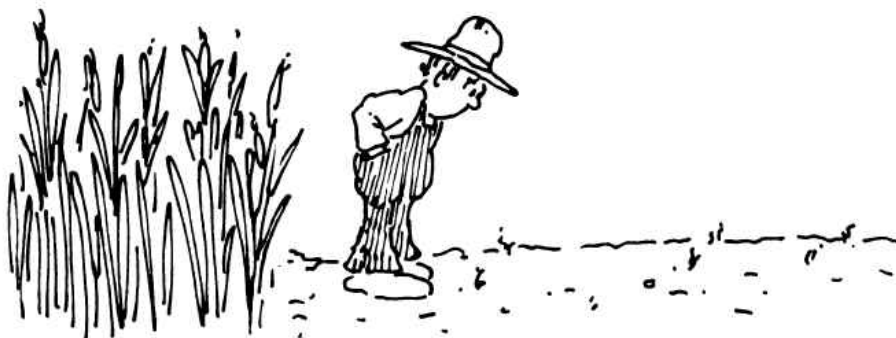
```
5 printchr$(14)
10 u=180000000
20 m=85000000
30 r1=1.0123
40 r2=1.0223
50 y=1960
60 u=u*r1
70 m=m*r2
80 y=y+1
90 ifm<uthen60
100 print"Anno =";y
110 printtab(8);"Popolazione "
120 print"U.S.A.   ";u
130 print"Mexico   ";m
```

```
Anno = 2037
        Popolazione
U.S.A.    461406538
Mexico    464464688
```

Ma che accadrebbe se la popolazione messicana stesse diligentemente tentando di mettere sotto controllo il tasso di natalità, riducendolo allo 0.001% all'anno, confrontato con il tasso americano dello 0.01%? Se così fosse, la popolazione messicana supererebbe mai quella statunitense?
Inserite le istruzioni 60, 70 e 85 nel primo programma, eseguitelo e trovate la risposta alla domanda.

```
5 printchr$(14)
10 u=180000000
20 m=85000000
30 r1=1.0123
40 r2=1.0223
50 y=1960
60 u=u*(1.01*r1)
70 m=m*(1.001*r2)
80 y=y+1
85 ify>9999then110
90 ifm<uthen60
100 print"Anno =";y
110 printtab(8);"Popolazione "
120 print"U.S.A.   ";u
130 print"Mexico   ";m
```

Ritornando al primo programma, eseguitelo con i dati della popolazione degli U.S.A. (180 milioni) e della California (15.7 milioni) dal 1960. A quell'epoca i tassi annui di crescita furono rispettivamente dell'1.23% e del 3.7%. L'esecuzione del programma indica l'anno in cui la popolazione della California supererà quella degli Stati Uniti. Questo, evidentemente, è assurdo. Dove è la contraddizione?
Questo programma avrebbe più senso se fosse reimpostato in modo da chiedere in quale anno la popolazione della California supererà quella del resto degli U.S.A., ammesso che ciò mai avvenga?
Il calcolo composto può essere usato anche per risolvere altri tipi di problemi attinenti la crescita della popolazione. Considerate quei vermi che si riproducono suddividendosi in 24 segmenti, ognuno dei quali rigenera una nuova testa ed una nuova coda. Qual è il numero massimo di vermi generabili in questo modo, partendo con un solo verme, dopo dieci divisioni? Supponendo che le divisioni avvengano ogni 22 giorni, quanti discendenti genererà un verme in un anno? Quando la terra sarà completamente coperta da questi animali?

Ecco un altro problema per il quale viene utile il principio del calcolo composto. La natura impiega circa 500 anni per produrre un pollice di terreno fertile superficiale. Molti anni fa gli Stati Uniti avevano una profondità media di quasi nove pollici di questa buona terra, ma nel 1975 si era scesi a circa sei pollici. Questo tipo di terra è ovviamente indispensabile per la crescita del cibo.
Una gestione disattenta della terra fa si che ogni anno circa l'1% sia eroso e poi perduto per sempre. Quando la profondità della terra raggiunge i 3 pollici o meno, è impossibile far crescere il raccolto su larga scala.
Scrivete un programma che calcoli l'anno in cui gli U.S.A. avranno meno di 3" di terra superficiale, supponendo che essa continui ad essere erosa al ritmo dell'1% annuo.
Sarete ancora vivi allora? Saranno vivi i vostri figli? Ci sarà ancora qualcuno in vita?

6
Probabilità

Le statistiche e la probabilità sono argomenti capaci di evocare ogni tipo di immagine. Alcune persone che non si sono divertite affatto al corso scolastico di statistica si tengono il più lontano possibile dall'argomento. Per altri, la statistica è quella cosa con cui i produttori disonesti possono fuorviarvi sulla qualità dei propri prodotti.

Non molto tempo fa una casa automobilistica estera si vantò che il 90% delle proprie auto vendute negli Stati Uniti negli ultimi sette anni era ancora su strada. Questo sembra un segno di eccellente affidabilità. Ma considerate il fatto che questo produttore si stava rapidamente espandendo nel mercato americano ed il 65% delle auto che esso aveva venduto negli U.S.A. erano state vendute negli ultimi due anni. C'è da aspettarsi che tutte queste ultime auto siano ancora in funzione.

Se il produttore avesse venduto nei primi 3 anni il 10% del totale venduto in sette anni e la maggior parte di queste auto non fossero più in circolazione, allora la pretesa di questa pubblicità perderebbe gran parte del suo significato.

La statistica e la probabilità non sono difficili se le si avvicina in modo logico, passo dopo passo. In realtà, possono anche essere molto divertenti.

Alcuni programmi usano una formula mentre altri simulano l'evento in cui sono coinvolti. I risultati sono gli stessi, ma le simulazioni possono aiutarvi a comprendere esattamente che cosa sta accadendo.

Il Triangolo di Pascal: metodo calcolato

Il Triangolo di Pascal (x) è decisamente affascinante. Come potete vedere dalla parte riprodotta qui sotto, ogni riga è simmetrica. Ogni riga contine anche i coefficienti per un'espansione binomiale. Le somme lungo le diagonali ascendenti formano la sequenza di Fibonacci. Le somme in orizzontale sono tutte potenze di 2. Ogni riga corrisponde alle cifre di una potenza di 11. Ogni elemento è la somma delle due cifre al di sopra. E, se la cosa vi interessa, tutti i suoi elementi sono identità nella teoria combinatoria.

I modi in cui questo meraviglioso triangolo può essere generato sono vari ed interessanti come le sue proprietà benché, forse, più difficili da comprendere. Ecco un modo per usare un computer per generare il triangolo.

Ogni elemento può essere trovato sommando insieme i due elementi sopra di esso. Il programma utilizza questo principio per produrre un triangolo con 8 livelli di profondità. Le linee 10-30 impostano a 1 la diagonale a destra.

Ogni elemento viene memorizzato nella variabile bidimensionale P(R,C), dove R rappresenta la riga e C la colonna. La variabile T (linea 50) lascia semplicemente alcuni spazi in modo tale che l'output assomigli ad un triangolo. Il punto cruciale del calcolo si trova alla riga 70, dove viene calcolato il valore di ogni nuovo elemento.

C'è un altro modo interessante per calcolare il Triangolo di Pascal con un numero di istruzioni addirittura minore di quelle del programma illustrato.

Esso genera il triangolo un elemento alla volta e non fa uso di alcun vettore o variabile bidimensionale. Riuscite ad immaginare come si possa scrivere un tale programma?

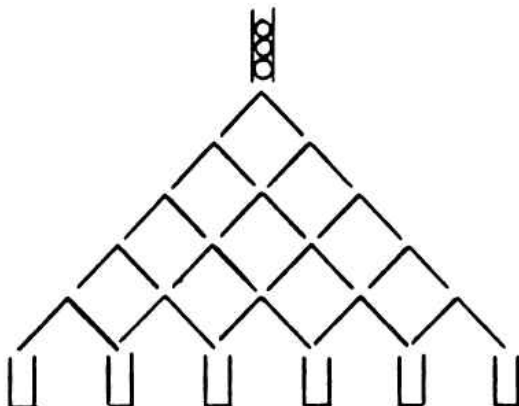
```
10 forc=1to8
20 p(c,c)=1
30 nextc
40 forr=1to8
50 t=20-r*2
60 forc=2tor+1
70 p(r+1,c)=p(r,c)+p(r,c-1)
80 ifp(r,c)=0then110
90 printtab(t);p(r,c);
100 t=t+4
110 nextc
120 print
130 print
```

```
                  1
               1     1
            1     2     1
         1     3     3     1
      1     4     6     4     1
   1     5    10    10     5     1
1     6    15    20    15     6     1
```

```
140 nextr
```

(*): In Italia, per evidenti ragioni storiche, lo stesso triangolo è più spesso detto "di Tartaglia" (N.d.T.).

Il Triangolo di Pascal: metodo probabilistico

Questo programma simula la caduta di una palla attraverso una struttura triangolare come quella illustrata qui sotto. Ad ogni livello la palla ha la stessa probabilità di cadere a destra o a sinistra. Un recipiente è posto alla base di ogni estremità inferiore; le palle si raccolgono in questi recipienti. Quando tutti i gruppi di palle sono stati fatti cadere, il numero di palle in ogni recipiente viene contato e visualizzato.

```
5 printchr$(14)
10 print"Triangolo di Pascal per probabilita'"
20 input"N. di palle  ";m
30 input"N. di livelli ";k
40 print"Posizione palle "
50 forn=1tom
60 t=0
70 forl=1tok
80 ifrnd(1)>.5then100
90 t=t+1
100 nextl
110 b(t+1)=b(t+1)+1
120 nextn
130 forl=1tok+1
140 printl;tab(12);b(l)
150 b(l)=0
160 nextl
```

```
Triangolo di Pascal per probabilita'
N. di palle  ? 1000
N. di livelli ? 2
Posizione palle
 1          264
 2          482
 3          254
```

```
Triangolo di Pascal per probabilita'
N. di palle  ? 1000
N. di livelli ? 4
Posizione palle
 1          58
 2          266
 3          395
 4          222
 5          59
```

```
Triangolo di Pascal per probabilita'     Triangolo di Pascal per probabilita'
N. di palle  ? 160                       N. di palle  ? 640
N. di livelli ? 4                        N. di livelli ? 6
Posizione palle                          Posizione palle
 1              12                        1              14
 2              39                        2              66
 3              62                        3              134
 4              36                        4              201
 5              11                        5              144
                                          6              73
                                          7              8
```

Il numero di palle depositatosi nei vari recipienti ai vari livelli dovrebbe avvicinarsi per approssimazione al corrispondente numero del Triangolo di Pascal se ridotto al minimo denominatore comune. Sapete il perché?
Lanciate diverse volte il programma variando il numero di palle. I numeri ottenuti approssimano davvero quelli del Triangolo di Pascal? Se, ad ogni biforcazione, ogni altra palla andasse nella direzione opposta, si otterebbe allora il Triangolo di Pascal. Provate ad eseguire il programma con 4 palle al livello 2, con 8 palle al livello 3 e con 16 palle al livello 4. I risultati ottenuti assomigliano a quelli del precedente programma che calcolava il triangolo? L'approssimazione migliora all'aumentare del numero di palle?
Come determinare la misura in cui i risultati così ottenuti si avvicinano al valore esatto del triangolo? Un modo consiste nel dividere il numero di palle cadute in ogni recipiente ad un dato livello per il numero totale di palle diviso per la somma teorica della riga. Perciò nel livello 4 dell'esempio, si divide il numero di palle in ogni recipiente per 62.5 (1000/16).
Poi si confronta il risultato con il valore "corretto" e si calcola la differenza percentuale.
Ecco il risultato di questa procedura per il livello 4 (in effetti la quinta riga) nell'esecuzione di prova.

Recipienti	*Palle*	*÷ 62.5*	*Corretto*	*Deviazione*
1	64	1.02	1	2.00%
2	241	3.86	4	3.50%
3	390	6.24	6	4.00%
4	241	3.86	4	3.50%
5	64	1.-2	1	2.00%

Compleanni comuni

Ecco un interessante piccolo problema riguardante la probabilità. In un gruppo di dieci persone selezionate casualmente, qual è la probabilità che qualcuno abbia lo stesso compleanno di un altro? E se il gruppo è di 20 persone? O di 50?
Al contrario, di quante persone deve essere costituito il gruppo perché ci sia il 50% di probabilità che almeno due di loro abbiano lo stesso compleanno? Quante persone sono necessarie per una probabilità del 90%?
Cercate di rispondere a queste domande, per intuizione o per calcolo, prima di guardare l'output del programma.
Questo programma costituisce un'introduzione indolore al mondo della statistica. Il calcolo è in realtà piuttosto banale. La probabilità che una persona qualunque in un gruppo abbia il compleanno il 1° Gennaio è di 1/365.
Se nel nostro gruppo vi sono solo due persone, la probabilità che entrambe compiano gli anni il 1° Gennaio è di 1/365 per 1/365, cioè un numero decisamente piccolo. La probabilità che abbiano un compleanno in comune è perciò pari a circa lo 0.27%.
Se, tuttavia, vi sono tre persone nel gruppo, la probabilità aumenta leggermente. Diciamo che le tre persone si chiamano Betsy, Ken e Larry. Dal ragionamento precedente sappiamo che c'è una probabilità dello 0.27% che Betsy e Ken abbiano lo stesso compleanno; lo stesso 0,27% che Betsy e Larry abbiano lo stesso compleanno ed infine lo stesso 0.27% per Larry e Ken. Quindi la probabilità totale per un gruppo di 3 persone è tre volte la probabilità per due sole persone.
Un gruppo di quattro aumenta la probabilità, rispetto al gruppo di due, di un fattore sei, cinque persone di un fattore 10, sei persone per 15, sette persone per 21, e così via. C'è qui una progressione, ma la probabilità può essere anche calcolata con la formula:

$$P=1 - (365\text{-}N)/365$$

Siete in grado di comprendere perché questa formula genera lo stesso risultato derivante dalla descrizione a parole e dalla progressione ad essa associata?
Prendete in considerazione tutti i presidenti degli Stati Uniti (quanti ce ne sono stati finora?). Due di loro, James Polk e Warren Harding, nacquero il 2 novembre. È questo il risultato che ci si poteva aspettare in base alla dimensione del gruppo? Poiché i compleanni possono essere predetti, almeno statisticamente, deve essere possibile predire allo stesso modo le morti.
È anche interessante notare che John Adams, James Monroe e Thomas Jefferson morirono tutti il 4 luglio. Millard Fillmore e William Taft morirono entrambi l'8 marzo. Qual è la probabilità di questo tipo di eventi?

```
5 printchr$(14)
10 print"Persone      Probabilita'"
20 q=364/365
30 forn=2to40
40 p=100*(1-q)
50 printn;tab(20);int(p*100+.5)/100;"%"
60 q=q*(365-n)/365
70 nextn
```

```
Persone    Probabilita'        21            44.37 %
 2               .27 %         22            47.57 %
 3               .82 %         23            50.73 %
 4              1.64 %         24            53.83 %
 5              2.71 %         25            56.87 %
 6              4.05 %         26            59.82 %
 7              5.62 %         27            62.69 %
 8              7.43 %         28            65.45 %
 9              9.46 %         29            68.1 %
 10              11.69 %       30            70.63 %
 11              14.11 %       31            73.05 %
 12              16.7 %        32            75.33 %
 13              19.44 %       33            77.5 %
 14              22.31 %       34            79.53 %
 15              25.29 %       35            81.44 %
 16              28.36 %       36            83.22 %
 17              31.5 %        37            84.87 %
 18              34.69 %       38            86.41 %
 19              37.91 %       39            87.82 %
 20              41.14 %       40            89.12 %
```

Ed ora tocca a voi scrivere un programma. Ecco un gioco da fare con i vostri amici. Voi scommettete che essi hanno preferenze simili sui colori. Ogni persona del gruppo mette un penny mentre voi — siccome siete così sicuri di voi stessi (e della probabilità) — mettete un importo in cents pari al numero di persone del gruppo. Se voi vincete, prendete tutta la posta in gioco, mentre se vincono i vostri amici, ognuno di loro raddoppia la sua scommessa originaria, ovvero due cents.
Poi ogni persona sceglie un colore (tra un elenco maggiore del numero di persone presenti, naturalmente). Se due di loro avranno scelto lo stesso colore, voi avrete vinto; se ognuno ha invece scelto un colore diverso, allora avrete perso.
Dato che voi volete vincere, dovete sapere quanti colori vanno messi sulla lista in base alle dimensioni del gruppo, per ottenere una possibilità di vincita superiore al 50%. Potreste volere ad esempio che la probabilità sia intorno al 70% circa. Potete generare la tabella necessaria con un programma di sole cinque linee. Avanti!

Le monete in tasca

I due prossimi programmi furono scritti originariamente da Glenda Lappan e M.J. Winter ed apparvero nella rivista Creative Computing. Si tratta di splendide simulazioni in grado di illustrare vari aspetti della probabilità.
Un giornale costa 5 cents. Un cliente ha in tasca 5 pennies ed un dime e paga il giornale chiedendo a voi, che siete l'edicolante, di scegliere a caso due tra le sei monete. Se ripetete questa procedura per tutti i 20 giorni lavorativi di un mese, quanto avrete ricavato in più o in meno rispetto ad $1.00 (20 giorni x 5 cents)?
Il seguente programma risolve questo piccolo problema. La prima moneta è scelta casualmente tra un gruppo di 6 (linea 50 in cui X = INT (RND*6)). Il valore di X può essere 0, 1, 2, 3, 4 o 5. Se è 0, consideriamo che sia stata estratta la dime e non facciamo l'altra estrazione, in quanto sarà sicuramente un penny. Perciò vengono aggiunti 11 cents al totale della linea 110.
Se la prima moneta è un penny (X=1, 2, 3, 4 o 5) allora facciamo una seconda estrazione tra le cinque monete rimanenti. Di nuovo, se viene scelta una dime (Y=0), vengono aggiunti 11 cents al totale; altrimenti si aggiungono 2 cents.
Come potete vedere dall'esempio, dopo un gran numero di tentativi, l'importo medio ricavato ogni giorno sembra avvicinarsi ai 5 cents. Se volete convincervi che la risposta è in effetti 5 cents, considerate il seguente ragionamento.
Date una lettera ad ogni moneta, da A ad F. Sia A la dime e da B a F i pennies; poiché tutte le combinazioni sono ugualmente probabili, tutte le possibili combinazioni sono:

AB
AC BC
AD BD CD
AE BE CE DE
AF BF CF DF EF

Ci sono cinque combinazioni con una dime (5*11 cents) e dieci di soli pennies (10*2 cents). Sommatele e otterrete 55 cents più 20 cents divisi per un totale di 15 combinazioni, che corrispondono ad un valore medio di 75/15, ovvero 5 cents.

```
5 printchr$(14)
10 print"Simula l'estrazione casuale di 2 monete"
20 input"N. prove ";n
30 u=0
40 fork=1ton
50 x=int(rnd(1)*6)
60 ifx=0then110
70 y=int(rnd(1)*6)
80 ify=0then110
90 u=u+2
100 goto120
110 u=u+11
120 nextk
130 print"Il valore medio e' ";u/n;" cent."
```

```
Simula l'estrazione casuale di 2 monete
N. prove ? 100
Il valore medio e'  5.24  cent.
```

```
Simula l'estrazione`casuale di 2 monete
N. prove ? 1000
Il valore medio e'  4.475  cent.
```

Le figurine di baseball

Nella statistica e nella probabilità è spesso necessario prevedere il numero di tentativi necessari in media per ottenere un risultato positivo. Per esempio, se state tentando di ottenere un quattro con un dado, quanti tentativi ci vorranno in media?
Siccome un dado ha sei lati, la probabilità di far uscire un numero qualunque tra 1 e 6 è P=1/6. Così, per ogni tiro, c'è una possibilità su sei che esca il numero quattro e 5 su 6 che esca un altro numero. Questa probabilità è detta Q. Quindi, per ottenere un quattro, abbiamo 1/6 di probabilità al primo tiro. Su due tiri, la probabilità è di due tentativi per la probabilità di un insuccesso, e poi il successo (2x5/6x1/6). Al terzo tiro, la probabilità di successo è di tre tentativi per la probabilità di due insuccessi seguiti da un successo (3x5/6x5/6x1/6).
Continuando con questo ragionamento si arriva alla formula che dà il numero di tentativi necessari per un successo:

$$E=1p+2q \times p+3q \times p+4q \times p+...$$

Questa successione può essere ridotta e risolta per E:

$$E = \frac{1}{1-q} = \frac{1}{p}$$

Così ora si può calcolare la risposta al problema originale. Il numero previsto di tentativi per ottenere un quattro è 1/p=6.
Ma con l'aiuto di un computer c'è un altro approccio a questo tipo di problema. Supponiamo che nelle confezioni di cereali vi siano dieci diversi premi. Quante confezioni di cereali si dovranno acquistare per avere la collezione completa? Possiamo espandere la formula precedente per risolvere questo problema:

$$\frac{N}{N} + \frac{N}{N-1} + \frac{N}{N-2} + \cdots + \frac{N}{1} = N(1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{N}$$

Finché il valore è dieci, questa formula è risolvibile a mano. Supponiamo però che vogliate risolvere questo problema per le figurine di baseball che si trovano nei pacchetti di gomme da masticare. Se una collezione completa è costituita da 50 figurine, quanti pacchetti di gomme da masticare dovete comprare, in media, per avere la collezione completa? Scrivete un programma utilizzando la formula generale sopradescritta. La risposta per 50 figurine è di 224.96 pacchetti; quanti pacchetti si devono acquistare per una collezione di 100 figurine?
C'è ancora un altro approccio al problema. Questo metodo è simile a quello usato per estrarre a caso le monete dalla tasca. In questo caso, il gruppo casuale è definito pari alla quantità totale di figurine; questo valore è letto in input all'istruzione 20. Ogni

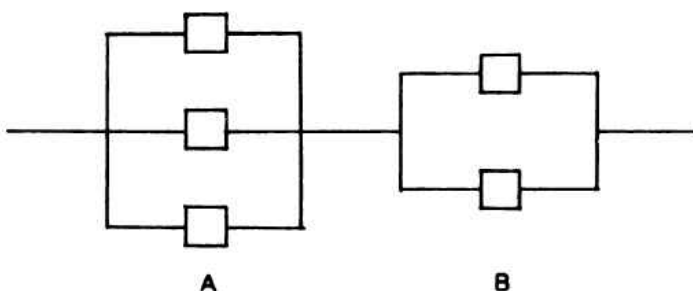
acquisto è posto pari ad un numero casuale compreso tra 1 e N (istruzione 120). Questa figurina è poi messa al suo posto nella collezione (istruzione 130). Se tuttavia in quella posizione c'è già una precedente figurina, incrementiamo semplicemente il contatore dell'acquisto (istruzione 160) ma senza avvicinarci al raggiungimento di una collezione completa. Dopo ogni acquisto verifichiamo se la collezione è completa (istruzione 170), altrimenti continuiamo ad acquistare altri pacchetti. Quando la collezione è completa, la quantità di pacchetti acquistati è totalizzata e scritta. Dopo una serie di tentativi viene calcolata la media. Questo valore medio dovrebbe essere ragionevolmente vicino al valore ottenuto con la formula, benché ci vogliano molti tentativi prima che i due numeri si discostino per meno dell'1%.

```
5 printchr$(14)
10 print"Simulazione dell'acquisto di gomma"
20 print"mediante il metodo delle figurine di    baseball."
21 input"Figurine in totale ";n
30 input"N. figurine uguali ";k
40 dimc(100)
50 print"Tentativi     Pacchetti"
60 s=0
70 fori=1tok
80 d=0
90 forj=1ton
100 c(j)=0
110 nextj
120 x=int(rnd(1)*n)+1
130 c(x)=c(x)+1
140 ifc(x)=1then160
150 goto120
160 d=d+1
170 ifd=nthen190
180 goto120
190 t=0
200 forj=1ton
210 t=t+c(j)
220 nextj
230 printi;tab(11);t
240 s=s+t
250 nexti
260 print"Media";s/k
```

```
Simulazione dell'acquisto di gomma
mediante il metodo delle figurine di    baseball.
Figurine in totale ? 10
N. figurine uguali ? 5
Tentativi    Pacchetti
 1             28
 2             22
 3             30
 4             18
 5             30
Media 25.6
```

Se eseguite questo secondo programma per una collezione di 100 figurine, impiegherà talvolta molto tempo prima di fornire una risposta. Provate con una collezione da 500 figurine, quantità usata in realtà da alcuni produttori di gomme da masticare. Potete andare a pranzare mentre il programma calcola soltanto il primo valore. Assicuratevi di aver modificato l'istruzione Dimension alla linea 40, in C(500).

Affidabilità di un sistema

Man mano che un numero sempre maggiore di persone nel mondo dipendono da strumenti meccanici ed elettronici, in una miriade di modi diversi, diviene sempre più importante che questi strumenti tecnologici continuino a funzionare.
Alcuni anni fa i militari istituirono una misura che poteva essere applicata ad ogni tipo di sistema, grande o piccolo, per misurarne l'affidabilità. Viene detta "tempo medio tra i guasti". Si esprime così quanto tempo passa, in media, tra il verificarsi di due guasti successivi. Per un carro armato, può essere anche soltanto di 100 ore, mentre per un velivolo spaziale l'MTBF (Medium Time Between Failures) deve essere considerevolmente maggiore della durata prevista per la missione.
Come abbiamo visto in una sezione precedente, è spesso auspicabile suddividere un grosso problema in sottoproblemi più piccoli. Calcolare l'MTRE per una navicella spaziale sarebbe proprio impossibile. È invece necessario incominciare con sistemi più piccoli e poi comporre il tutto.
Considerate uno dei sottosistemi elettrici di una navicella spaziale. Esso utilizza cinque componenti, come illustrato qui sotto, due sistemi paralleli A e B, fra loro in serie. I sottosistemi paralleli sono detti "ridondanti".
Questo è un metodo per aumentare l'affidabilità, in quanto il sistema continuerà a funzionare se funziona almeno uno dei componenti.

Se il produttore dei componenti dichiara che ognuno di essi ha il 60% di probabilità di durare 1000 ore, qual è la possibilità che l'intero sistema duri 1000 ore? Provate a indovinare, prima di leggere la soluzione. La vostra ipotesi è maggiore o minore di 60%? Perché?
Il programma seguente è una simulazione di questo sistema. Ricordate che il sottosistema A continuerà a funzionare se uno qualunque dei tre componenti è in funzione, ed il sottosistema B funziona purché funzioni almeno uno dei suoi due componenti. Il sistema si fermerà invece se tutti e tre i componenti di A oppure entrambi i componenti di B si guastano.
Nell'istruzione 30, il programma viene impostato in modo che effettui 500 prove sul sistema. L'istruzione 50 estrae a caso un numero compreso tra 1 e 10 per ogni

componente in ogni prova. Se il valore di questo intero è 6 o meno, il componente è a posto (60% di probabilità di funzionamento al termine delle 100 ore). Se è invece 7, 8, 9 oppure 10, ciò significa che il componente si è guastato ed il programma scrive allora "Guasto!".

```
5 printchr$(14)
10 print"Simula l'uso di un circuito per 1000 ore":print:print
20 s=0
30 fori=1to500
40 forj=1to5
50 c(j)=int(rnd(1)*10)+1
60 ifc(j)>6then90
70 c(j)=1
80 goto100
90 c(j)=0
100 nextj
110 ifc(1)=1orc(2)=1orc(3)=1then140
120 print"Guasto !";
130 goto180
140 ifc(4)=1orc(5)=1then170
150 print"Guasto !";
160 goto180
170 s=s+1
180 nexti
190 print:print:print"L' affidabilita' e' circa ";int(10000*(s/500)+.5)/100;"%"
```

```
Simula l'uso di un circuito per 1000 ore

Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto
Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto
Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto
Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto
Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto
Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto
Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto
Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto
Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto
Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto
Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto
Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto
Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !Guasto !

L' affidabilita' e' circa  74.2 %
```

L'istruzione 110 verifica se il sottosistema A è in funzione, mentre l'istruzione 140 controlla il sottosistema B. Se entrambi funzionano, il successo è registrato nell'istruzione 170.

La stessa cosa si può ovviamente fare anche con metodi più veloci. Ad esempio tutte le singole verifiche (dalla linea 110 alla 140) possono essere sostituite da un unico test complessivo (nuova istruzione 110).

```
110 if (c(1)+c(2)+c(3)))*(c(4)+c(5))=0 then 140
130 s=s+1
140 next i
```

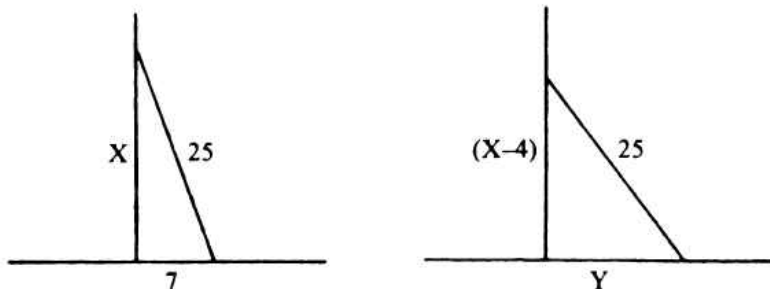
```
L' affidabilita' e' circa  78.9 %
```

Ora tocca a voi fare uso di quanto avete imparato in questo capitolo e scrivere un programma che determini l'MTBF del sistema prima descritto.

7
Geometria e calcoli

In questo capitolo vengono usati, per risolvere problemi geometrici, molti degli approcci alla soluzione di problemi, presentati nei capitoli precedenti.
Scoprirete che molti problemi possono essere risolti in una gran varietà di modi diversi — applicando una formula, per tentativi ed errori, per approssimazioni successive e, in certi casi, con il buon senso. Forse la cosa più importante che si può imparare da questi problemi e programmi è come analizzare un problema in modo da raggiungere la soluzione velocemente e facilmente.

I problemi delle scale incrociate e che scivolano

Eccovi un semplice problema relativo ad una scala che scivola. Una scala lunga 25 piedi è appoggiata al muro in modo tale che la sua base dista 7 piedi dal muro stesso. La base della scala scivola sulla ghiaia in modo che la sua sommità si abbassa di 4 piedi rispetto alla sua posizione iniziale. Di quanto è scivolata la base della scala?

I diagrammi mostrano le due posizioni della scala. Sappiamo, dal teorema di Pitagora, che $a^2+b^2=c^2$, perciò le equazioni atte a risolvere il problema sono:

$$x = \sqrt{25^2 - 7^2}$$

$$y = \sqrt{25^2 - (x-4)^2}$$

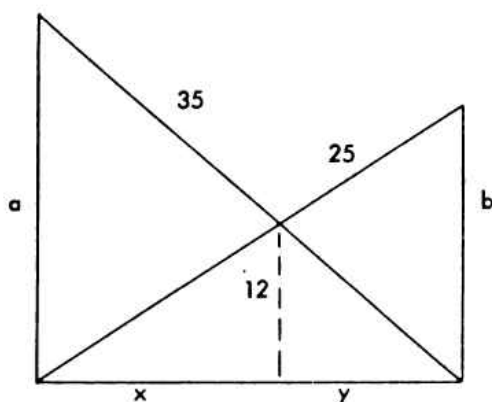
e lo scivolamento della base è allora $z=y-7$. Mettendo queste tre equazioni in un programma, otteniamo rapidamente la risposta di 8 piedi.

```
5 print chr$(14)
10 print "La scala che scivola"
20 c=25
30 b=7
40 x=sqr(c↑2-b↑2)
50 y=sqr(c↑2-(x-4)↑2)
60 z=y-b
70 print "La base e' scivolata di";z;"ft"
```

```
La scala che scivola
La base e' scivolata di 7.99999999 ft
```

Benché il problema non comporti grosse difficoltà aritmetiche, non è molto divertente risolverlo a mano. Per valori molto più complessi si può ben apprezzare la capacità del computer, ad esempio con una lunghezza di 27.83 piedi ed una distanza dal muro di 7.62 piedi.

```
5 print chr$(14)
10 print "La scala che scivola"
20 c=27.83
30 b=7.62
40 x=sqr(c↑2-b↑2)
50 y=sqr(c↑2-(x-4)↑2)
60 z=y-b
70 print "La base e' scivolata di";z;"ft"

La scala che scivola
La base e' scivolata di 8.38613153 ft
```

Consideriamo ora un vecchio problema, presente in molte raccolte classiche. Due scale, una lunga 25 piedi e l'altra 35, sono appoggiate su due edifici di un vicolo, l'una di fronte all'altra, come mostrato nel seguente disegno. Le due scale si incrociano in un punto posto a 12 piedi di altezza rispetto a terra. Quanto è largo il vicolo?

Utilizzando due volte la similitudine dei triangoli, troviamo che:

$$12/a + 12/b = 1$$

Applicando poi il teorema di Pitagora e successivamente riducendo, otteniamo:

$$a^2 - b^2 = 600$$

Usando uno dei metodi descritti nel capitolo sulla risoluzione di problemi, potete risolvere queste due equazioni simultanee. Oppure possono essere combinate in un'unica equazione in cui l'ampiezza del vicolo (z) è:

$$z = \sqrt{35^2 - a^2}$$

Eliminando b, si ottiene la seguente equazione di quarto grado:

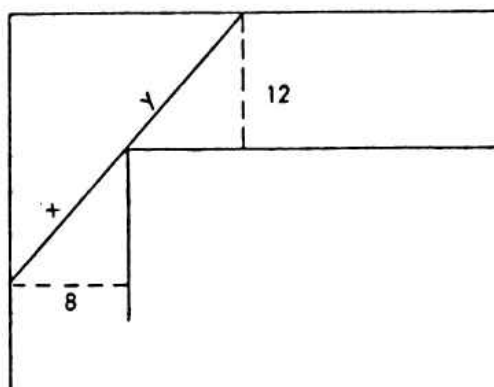
$$a^4 - 24a^3 - 600a^2 + 14400a - 86400 = 0$$

Anche questa può essere risolta con uno tra i metodi esposti nel suddetto capitolo. C'è tuttavia un approccio alternativo a questo metodo tradizionale.

La soluzione è l'intersezione delle curve descritte dalle due equazioni originali. Con successive approssimazioni, ad esempio incrementando a di 1 alla volta, si possono trovare b_1 e b_2 nelle seguenti equazioni, derivanti dalle originali:

$$b_1 = \frac{12a}{a - 12}$$

$$b_2 = \sqrt{600 - a^2}$$

Quando b_1 scende al di sotto di b_2, riducete l'incremento a 0.1 per ottenere una migliore approssimazione. Continuando con questo metodo di approssimazioni successive, è possibile ottenere una soluzione molto accurata.
C'è forse qualche altro approccio? Sì, c'è ed evita anche l'equazione di quarto grado. Utilizza infatti le equazioni originali in una procedura per tentativi ed errori, come descritto nel relativo capitolo. Provate a scrivere un programma che sfrutti questo approccio.
Ed ecco infine un altro classico problema che potete risolvere nel modo che preferite. Qual è la lunghezza massima di una scala che possa essere trasportata in posizione orizzontale, in modo da farla passare girando un angolo ottenuto dall'incrocio di un vicolo largo 12 piedi con uno largo 8 piedi? Il problema è meglio comprensibile osservando il seguente diagramma.

Distanza tra punti espressi in coordinate

Questo programma determina la distanza tra due punti qualsiasi, la cui posizione nello spazio tridimensionale è espressa in base alle coordinate x, y e z.
La formula relativa alla distanza tra due punti nello spazio tridimensionale è:

$$d = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2$$

Sarebbe opportuno che il programma fosse in grado di risolvere qualunque problema di questo tipo. Un approccio potrebbe consistere nell'utilizzo di istruzioni INPUT per richiedere le coordinate dei punti. Un altro approccio utilizza istruzioni DATA per definire i punti. Così per impostare un nuovo problema è sufficiente modificare questa istruzione.
Il programma è scritto in modo da calcolare la distanza fra tre serie di punti. I punti utilizzati sono:

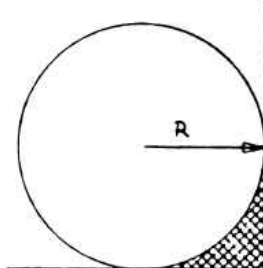
0, 0, 0	e	3, 4, 5
3.5, -4.7, 6.2	e	-0.9, 3.0, 4.4
67, 36, 82	e	54, 25, 90

Questo tipo di calcolo è molto usato nella navigazione aerospaziale. Riuscite a determinare l'angolo della risultante traiettoria di volo? Vi consigliamo di incominciare con due sole dimensioni.

```
5 print chr$(14)
10 print chr$(147)
20 print "Calcola la distanza tra punti          nello spazio 3D"
30 print
40 print "        Coordinate","        Distanza":print
50 read a,b,c,d,e,f
60 l=sqr((a-d)↑2+(b-e)↑2+(c-f)↑2)
70 print a,b,c
80 print d,e,f,int(l*1000+.5)/1000
90 print:print
100 goto 40
110 data 0,0,0,3,4,5
120 data 3.5,-4.7,6.2,-.9,3,4.4
130 data 67,36,82,54,25,90
```

```
Calcola la distanza tra punti          nello spazio 3D

      Coordinate                          Distanza

 0           0           0
 3           4           5            7.071

      Coordinate                          Distanza

 3.5         -4.7            6.2
-.9           3           4.4            9.049

      Coordinate                          Distanza

 67           36          82
 54           25          90             18.815

      Coordinate                          Distanza
```

Area - metodo calcolato

È molto semplice calcolare l'area di figure geometriche regolari, utilizzando le consuete formule. Il problema diviene tuttavia più difficile quando si voglia calcolare l'area derivante dalla combinazione di due forme regolari oppure l'area di forme irregolari. Questo programma mostra il metodo di analisi atto a risolvere un problema del primo tipo, mentre il secondo programma mostra quattro metodi per il calcolo di aree irregolari.
Il problema consiste nel determinare l'area ombreggiata nella seguente figura per qualunque valore del raggio R.

Il primo passo richiede l'utilizzo delle formule per calcolare l'area di un cerchio e di un quadrato:

$$A(\text{cerchio}) = \pi * R^2$$
$$A(\text{quadrato}) = \text{Lato}^2 \text{ oppure } (2*R)^2$$

Possiamo così calcolare la differenza tra l'area del quadrato e quella del cerchio ad esso inscritto:

$$A(\text{differenza}) = A(\text{quadrato}) - A(\text{cerchio})$$

e l'area di un angolo è quindi questa differenza divisa per 4. Il programma qui descritto è in grado di calcolare l'area per qualunque raggio.
Ed ora tocca a voi. Calcolate l'area della differenza tra un cerchio ed il quadrato in esso inscritto. Modificate poi il vostro programma per calcolare l'area della differenza tra un cerchio ed un triangolo, un esagono ed un ottagono rispettivamente inscritti nel cerchio stesso.

```
5 printchr$(14)
10 print"Area del cerchio inscritto nel          quadrato"
20 input"raggio";r
30 c=3.14159*r↑2
40 s=(2*r)↑2
50 d=s-c
60 d1=d/4
70 print"differenza";d
80 print"un angolo";d1
```

```
Area del cerchio inscritto nel quadrato
raggio ? 1
differenza .858409999
un angolo .2146025

Area del cerchio inscritto nel quadrato
raggio ? 10
differenza 85.8410001
un angolo 21.46025
```

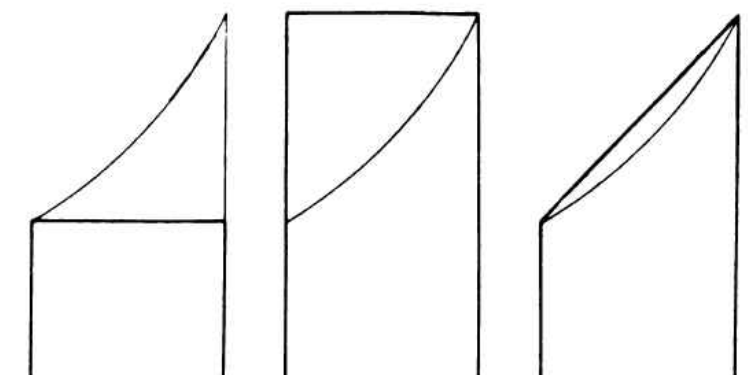
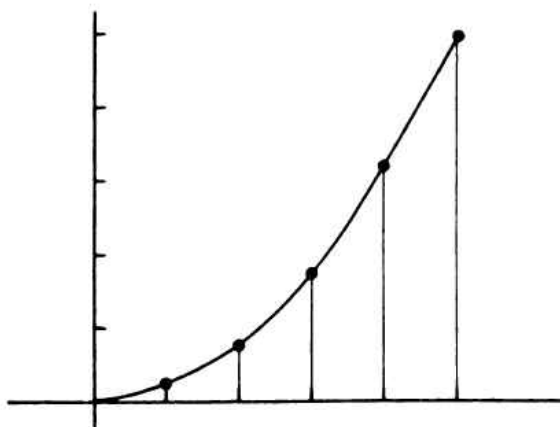
Estendete infine il vostro programma in modo che calcoli l'area della differenza tra ogni figura regolare inscritta in un cerchio di raggio 1.0. Costruite una tabella contenente i risultati, come la seguente:

Numero di lati	*Area della differenza*
3	
4	
5	
6	
7	
8	

Che cosa potete concludere da questi risultati? C'è una qualche progressione nei valori di queste aree?

Area - metodo per integrazione

In molti casi è necessario determinare l'area di una figura irregolare o l'area al di sotto di una curva per la quale non sia disponibile una formula esatta. L'approccio più generalmente usato consiste nel dividere l'area in piccoli intervalli regolari e poi sommare le aree di tutti questi pezzi.

La forma più facile da usare in questi calcoli è un rettangolo. Un gruppo di rettangoli può essere sia inscritto all'interno della figura rettangolare o della curva, sia circoscritto. I primi due diagrammi mostrano questi due metodi utilizzati per trovare l'area al di sotto di una curva. Un terzo metodo consiste nell'uso di trapezoidi i quali, a seconda della direzione della curvatura, vengono automaticamente inscritti oppure circoscritti.

Un quarto metodo, noto come Regola di Simpson, adatta essenzialmente una serie di parabole fra i punti della curva e calcola l'area media. Richiede che l'area sia divisa in un numero pari di strisce parallele. Chiamiamo 2m il numero totale di divisioni, poste a distanza h. Il punto sulla curva dove vi è intersezione con la prima linea è y0, il secondo y1, e così via fino a y_{2m}. L'area si ottiene allora con la formula:

$$A = \frac{1}{3}h[(y_0 + y_{2m}) + 4(y_1 + y_3 + \ldots + y_{2m-1}) + 2(y_2 + y_4 + \ldots y_{2m-2})]$$

L'area così calcolata converge in modo estremamente rapido rispetto agli altri metodi. Ciononostante, con l'aumentare del numero di intervalli tutti i metodi giungono allo stesso limite. Come era facile prevedere, il metodo trapezoidale converge più rapidamente di ogni altro metodo basato sui rettangoli. Calcolate la media tra i due metodi basati sui rettangoli. Che cosa ottenete? Questo risultato vi suggerisce forse un altro metodo?
I metodi utilizzati in questo programma coinvolgono dei calcoli. E voi pensavate che il calcolo fosse una cosa difficile! Ora sapete che non è cosi.

```
5 printchr$(14)
10 printchr$(147)
20 print"  Calcoli di una curva con 4 metodi di                integrazione"
40 print
50 print"Inserire il valore di inizio e di fine  per la x (il primo piu' piccolo
)"
60 inputa,b
70 print"intervallo della x=";a;",";b
80 print:print" No. intervalli"
90 print" Rettangoli inscr.      Trapezoidi
100 print" rettangoli circ.       Parabole"
110 m=-2,
120 s=0
130 deffna(x)=x↑3
140 m=m+3
150 forn=mtom+2
160 c=s
170 q=0
180 p=0
190 d=2↑n
200 h=(b-a)/d
210 fori=0tod-1
220 x=a+i*h
230 p=p+h*fna(x)
240 q=q+h*fna(x+h)
250 next i
260 t=(p+q)/2
270 u=fna(a)+fna(x+h)
280 forj=2to(d-2)step2
290 u=u+2*fna(a+j*h)
300 nextj
310 v=0
320 fork=1to(d-1)step2
330 v=v+4*fna(a+k*h)
340 nextk
350 s=(u+v)*(h/3)
360 print:printd:print:printp;tab(23);t:printq;tab(23);s
370 nextn
```

```
380 ifd<64then140
390 ifabs((c-s)/((c+s)/2))>.0001then140
400 stop
```

```
  Calcoli di una curva con 4 metodi di integrazione

Inserire il valore di inizio e di fine  per la x (il primo piu' piccolo)
 ? 1,10
intervallo della x= 1 , 10

 No. intervalli
 Rettangoli inscr.        Trapezoidi
 rettangoli circ.         Parabole

 2

 753.187501                3000.9375
 5248.6875                5499.75001

 4

 1501.17188                2625.04688
 3748.92188                2499.75

 8

 1969.13672                2531.07422
 3093.01172                2499.75

 16

 2226.61231                2507.58106
 2788.54981                2499.75

 32

 2361.22339                2501.70776
 2642.19214                2499.75

 64

 2429.99726                2500.23944
 2570.48163                2499.75

break in 400
```

8
Scienze

Utilizzando tecniche ed approcci presentati nei capitoli precedenti questo capitolo contiene cinque programmi nel campo scientifico. Uno utilizza una formula per risolvere semplici problemi relativi ai gas, due sono esercitazioni pratiche sulle leggi di Boyle e Charles, e gli ultimi due sono simulazioni.
Vedrete che le simulazioni si basano su molte fra le precedenti tecniche come le progressioni ed i calcoli ripetitivi.

Il volume dei gas

Eccovi un semplice programma in grado di produrre una tabella non altrettanto semplice di valori per i volumi dei gas.
Il volume di un gas varia in maniera direttamente proporzionale alla temperatura assoluta T (in gradi Kelvin) ed inversamente alla pressione P. Se una certa quantità di gas occupa 500 piedi cubi alla pressione di 53 libbre per piede quadrato ed una temperatura assoluta di 500 gradi, che volume occuperà a 600 gradi di temperatura assoluta e con una pressione che varia da 100 a 1000 libbre per piede quadrato, ad incrementi di 50 libbre?
Le condizioni originali sono usate per determinare la costante K nell'istruzione 50 ($K=V*P/T$). I nuovi volumi sono poi calcolati al variare della pressione con T pari a 600 gradi. La formula usata è $V=K*T/P$.
Nella seconda parte del programma, le linee da 70 a 100 vengono modificate in modo da produrre una curva del volume del gas alle varie pressioni.
Come modifichereste il programma per trattare un caso più generale (altri gas a temperature differenti)? In secondo luogo, riuscite a scrivere un programma che produca una tabella di valori per un gas a pressioni e temperature differenti?

```
5 printchr$(14)
10 print"Volume di un gas a  pressioni differenti"
20 v=500
30 p=53
40 t=500
50 k=v*p/t
60 input"Temperatura (k)";t1
70 print"Pressione  Volume"
80 forp=100to1000step50
90 v=k*t1/p
100 printp;tab(10);v
110 nextp
```

```
Volume di un gas a  pressioni differenti
Temperatura (k) ? 600
Pressione  Volume
 100       318
 150       212
 200       159
 250       127.2
 300       106
 350       90.8571429
 400       79.5
 450       70.6666667

 500       63.6
 550       57.8181818
 600       53
 650       48.9230769
 700       45.4285715
 750       42.4
 800       39.75
 850       37.4117647
 900       35.3333333
 950       33.4736842
 1000       31.8
```

```
5 printchr$(14)
10 print" Volume di un gas a pressioni diverse"
20 v=500
30 p=53
40 t=500
50 k=v*p/t
60 input"Temperatura (k)";t1
65 print
70 print"pressione   volume"
80 forp=100to1000step50
90 v=k*t1/p
100 printp;tab(4+v/17);"*"
110 nextp
```

```
 Volume di un gas a pressioni diverse
Temperatura (k) ? 600

pressione   volume
 100                   *
 150                *
 200              *
 250             *
 300            *
 350           *
 400          *
 450          *
 500         *
 550         *
 600         *
 650        *
 700        *
 750        *
 800        *
 850        *
 900        *
 950       *
 1000       *
```

Esercitazione sulla legge di Charles

Nel programma precedente, i volumi dei gas erano calcolati utilizzando la legge di Charles e la legge di Boyle. Ecco un programma di esercitazione pratica (ricordate il capitolo 1?) che produce problemi relativi al volume ed alla temperatura di un gas. A pressione costante, il rapporto volume/temperatura può essere espresso come segue:

$$\frac{V_0}{T_0} = \frac{V_1}{T_1}$$

Il programma presenta quattro problemi, uno per ciascuna delle quattro variabili dell'equazione precedente.
Come si può rendere più efficiente il programma? E più interessante?

```
5 printchr$(14)
10 print"Legge di Charles"
20 print"Volume in millilitri"
30 print"Temperatura in gradi Kelvin"
40 print
50 v=int(rnd(1)*50+50)*100
60 v1=int(rnd(1)*50+50)*100
70 t=int(rnd(1)*125+125)
80 t1=int(rnd(1)*125+125)
90 c=c+1
100 oncgoto110,150,190,230,270
110 v=0
120 print:print"Calcolare v per"
130 q1=v1*t/t1
140 goto280
150 v1=0
160 print"Calcolare v1 per"
170 q1=v*t1/t
180 goto280
190 t=0
200 print:print"calcolare t per"
210 q1=v*t1/v1
220 goto280
230 t1=0
240 print:print"Calcolare t1 per"
250 q1=t*v1/v
260 goto280
270 stop
280 print"v=";v;tab(11);"t=";t
290 print"v1=";v1;tab(11);"t1=";t1
300 input"Risposta";q
310 print"Valore esatto";q1
320 print
330 goto50
```

```
Legge di Charles
Volume in millilitri
Temperatura in gradi Kelvin


Calcolare v per
v= 0              t= 146
v1= 9300              t1= 220
Risposta 10084
Valore esatto 6171.81818

Calcolare v1 per
v= 7100               t= 241
v1= 0              t1= 134
Risposta 18439
Valore esatto 3947.71784


calcolare t per
v= 9500               t= 0
v1= 6400               t1= 152
Risposta 3034
Valore esatto 225.625


Calcolare t1 per
v= 9800               t= 205
v1= 9400               t1= 0
Risposta 16467
Valore esatto 196.632653
```

Esercitazione sulla legge di Boyle

La legge di Boyle descrive il comportamento dei gas in condizioni ideali al variare della pressione e del volume. Boyle scoprì che a temperatura costante

$$P_0 * V_0 = P_1 * V_1$$

La pressione è normalmente misurata in centimetri di mercurio mentre il volume è misurato in millilitri. Come per l'esercitazione sulla legge di Charles, questo programma presenta quattro problemi, uno per ciascuna variabile della equazione precedente.
A differenza di molti altri programmi analoghi, questi due non confrontano la vostra risposta con quella esatta. Lasciano invece che siate voi a farlo.
È una cosa auspicabile o no? Perché? Se per voi non è desiderabile, modificate i programmi in modo che confrontino le risposte e calcolino un punteggio.

```
5 printchr$(14)
10 print"Legge di Boyle"
20 print"Volume in millilitri"
30 print"Pressione in cmmer"
40 print
50 v=int(rnd(1)*50+50)*100
60 v1=int(rnd(1)*50+50)*100
70 p=int(rnd(1)*150+150)
80 p1=int(rnd(1)*150+150)
90 c=c+1
100 oncgoto110,150,190,230,270
110 v=0
120 print:print"Calcolare v per"
130 q1=p1*v1/p
140 goto280
150 v1=0
160 print"Calcolare v1 per"
170 q1=p*v/p1
180 goto280
190 p=0
200 print:print"Calcolare p per"
210 q1=p1*v1/v
220 goto280
230 p1=0
240 print:print"Calcolare p1 per"
250 q1=p*v/v1
260 goto280
270 stop
280 print"v=";v;tab(11);"p=";p
290 print"v1=";v1;tab(11);"p1=";p1
300 input"Risposta";q
310 print"Valore esatto";q1
320 print
330 goto50
```

```
Legge di Boyle
Volume in millilitri
Pressione in cmmer


Calcolare v per
v= 0              p= 200
v1= 9100              p1= 249
Risposta ?  452
Valore esatto 11329.5

Calcolare v1 per
v= 7600               p= 289
v1= 0               p1= 184
Risposta ?  329
Valore esatto 11936.9565


Calcolare p per
v= 9800               p= 0
v1= 9100               p1= 229
Risposta ?  8362
Valore esatto 212.642857


Calcolare p1 per
v= 6100               p= 262
v1= 7200               p1= 0
Risposta ?  8316
Valore esatto 221.972222
```

Emissioni fotoelettriche

Quando la luce a bassa lunghezza d'onda colpisce una superficie metallica, il metallo emette degli elettroni. In base alla descrizione di questo fenomeno fatta da Einstein, c'è una lunghezza d'onda massima per ogni metallo oltre la quale non vengono più emessi elettroni. Essa è chiamata lunghezza d'onda critica del metallo.
Questo programma simula un esperimento di laboratorio in cui un metallo è posto nel vuoto e bombardato con deboli raggi X. La quantità di elettroni emessi è misurata da un micrometro. Il programma simula tre prove per ognuna tra 9 lunghezze d'onda. Dopo ogni serie di dati sperimentali, il programma chiede se volete un'altra elaborazione con una maggiore intensità luminosa. Il motivo è che talvolta, a bassa intensità luminosa, non vengono emessi abbastanza elettroni perché si possa effettuare una misurazione significativa.
Potete aumentare la precisione dell'esperimento aumentando il numero di lunghezze d'onda utilizzate. Questo valore può essere modificato nell'istruzione 60; osservate che la variabile L viene divisa per 1000 per esprimere la lunghezza d'onda in Angstrom. Un Angstrom (A) è pari a 10^8 centimetri, o 10^{-4} micron.

Ecco i coefficienti per diversi metalli:

Argento	.308
Bismuto	.338
Cadmio	.318
Piombo	.340
Platino	.385

Difficilmente oggi un laboratorio di fisica di una scuola possiede gli strumenti sperimentali per eseguire questo esperimento, tuttavia l'equipaggiamento necessario può essere simulato con un piccolo computer. Ci sono molte altre cose che non potreste normalmente sperimentare, che possono essere invece simulate con un computer. Cose come un impianto nucleare, un'epidemia di malaria, un sistema di trasporto urbano di massa, una fabbrica di biciclette.
Siete capaci di scrivere una simulazione per un sistema reale? Troverete sezioni dedicate alla simulazione nei libri *Computers in Mathematics* e *Computer in Science and Social Studies.* Questi libri, insieme ad articoli apparsi su *Creative Computing*, possono essere d'aiuto per scrivere una vostra simulazione.

```
5 printchr$(14)
10 print"Bombardamento di un metallo con raggi x"
20 input"Coefficente";v
30 k=int(1+2*rnd(1))
35 print
40 print"Output in microampere"
50 print"Lunghezza   Prova       Prova"
55 print"d'onda       1            2"
60 forl=.42to.25step-.02
70 m=int(1000/l)
80 printm;
90 forj=1to2
100 ifl>vthen130
110 i=sqr(int(25*rnd(1)))
120 goto140
130 i=sqr(k*k*100+int(35*rnd(1)))
140 n=int(10*i+.5)/10
150 printtab(j*11);n;
160 nextj
170 print
180 nextl
190 print:input"aumenta l'intensita' della luce (s,n)";a$
210 ifa$="n"ora$="N"then270
220 input"Di che fattore (da 1 a 10)";f
240 k=k*f
250 print
260 goto50
270 stop
```

```
Bombardamento di un metallo con raggi x
Coefficente ? .34
Output in microampere
Lunghezza  Prova      Prova
 d' onda     1          2
---------  -----      -----
  2380      10.3      10.2
  2500      10.3      10.9
  2631      11.2      11
  2777      11        11.1
  2941      3.3       2.8
  3125      3.6       4.2
  3333      1         4.9
  3571      2.4       1.7
  3846      3.5       1.7


aumenta l'intensita' della luca (s,n) ? s
Di che fattore (da 1 a 10) ? 5


Lunghezza  Prova  Prova
---------  -----  -----
  2380      50.1   50
  2500      50.2   50.3
  2631      50.1   50.1
  2777      50.3   50.3
  2941      4.7    2.4
  3125      4.1    3.5
  3333      3.9    4.7
  3571      2      4.6
  3846      1.7    1.7


aumenta l'intensita' della luce (s,n) ? n
```

La mutazione delle farfalline

Questo programma simula la crescita di una colonia di farfalline. Il programma consente di introdurre una mutazione genetica entro un numero di anni compreso tra 1 e 30. La mutazione può favorire sia le farfalline chiare che quelle scure.
Il programma, così come è qui indicato, incomincia con una colonia di sole farfalline chiare; potreste aggiungere un gruppo iniziale di farfalline colorate nella linea 45 come P1.
L'esempio mostra una mutazione a favore delle farfalline scure, che avviene nell'anno terzo. La mutazione coinvolge circa il 2% delle farfalline chiare ogni anno e le fa diventare scure. Il programma mostra il numero di farfalline di ogni colore per un periodo di 30 anni.
Ovviamente, a causa del lungo periodo di tempo coinvolto, sarebbe difficile condurre questo esperimento a scuola, perciò il computer è ancora di reale aiuto.

```
5 printchr$(14)
10 print"Mutazione delle farfalline"
20 input"Tasso di mutazione (1,10)";m
30 p0=10000
40 p0=10000
50 z=p0
60 input"Anno in cui avviene la mutazione";x
80 l=1
90 d=2
100 input"Cambiamento in chiare o scure (l,d)";e$
120 print:print"Anno  Scure  Chiare"
125 print"----  -----  ------"
130 fort=1to30
140 ift>=xthen170
150 p1=0
160 goto230
170 ife$<>"l"ande$<>"d"then150
180 p1=int(p1+.01*m*p0+.5)
190 p0=int(z-p1+.5)
200 ifp1<zthen230
210 p1=z
220 p0=0
230 printt;tab(5);p1;tab(12);p0
240 nextt
```

Le cosiddette "api assassine", che furono introdotte in Sudamerica verso la metà degli anni '70 avrebbero dovuto costituire un aiuto per la produzione di miele perché erano molto più piene di energia che non le pigre api brasiliane.
L'idea era che si sarebbero accoppiate con le api esistenti generando una stirpe più produttiva. Incominciarono invece ad uccidere furiosamente la gente, terrorizzando il paese. Poi incominciarono a migrare verso nord. I funzionari per l'agricoltura degli U.S.A. temettero che le api potessero invadere il paese, seminando morte e distruzione.

```
Mutazione delle farfalline
Tasso di mutazione (1,10) ? 2
Anno in cui avviene la mutazione ? 3
Cambiamento in chiare o scure (l,d) ? d
```

Anno	Scure	Chiare
1	0	10000
2	0	10000
3	200	9800
4	396	9604
5	588	9412
6	776	9224
7	960	9040
8	1141	8859
9	1318	8682
10	1492	8508
11	1662	8338
12	1829	8171
13	1992	8008
14	2152	7848
15	2309	7691
16	2463	7537
17	2614	7386
18	2762	7238
19	2907	7093
20	3049	6951
21	3188	6812
22	3324	6676
23	3458	6542
24	3589	6411
25	3717	6283
26	3843	6157
27	3966	6034
28	4087	5913
29	4205	5795
30	4321	5679

Anno dopo anno, le api assassine incominciarono ad accoppiarsi con le più docili api sudamericane, ma solo al tasso del 3% all'anno. Supponendo che esse possano costituire un pericolo per gli U.S.A. finché il loro numero non sia ridotto al 15% della loro quantità originaria, quanti anni ci vorranno perché ciò avvenga? Gli attuali modelli di migrazione le condurranno negli Stati Uniti entro il 1989; saranno ancora pericolose (considerando che furono introdotte nel 1974)?

Il moto di un proiettile

Il percorso seguito da un proiettile è detto la sua traiettoria. La traiettoria è influenzata notevolmente dalla resistenza dell'aria, il che rende estremamente complesso fare un'analisi esatta del moto. Noi tuttavia trascureremo gli effetti della resistenza dell'aria, immaginando che il moto abbia luogo nel vuoto.
Nel caso generale del moto di un proiettile, il corpo (pallottola, razzo, ecc.) possiede una velocità iniziale con un angolazione θ sopra (o sotto) lo orizzonte.
Se V_0 rappresenta la velocità iniziale, le componenti orizzontale e verticale sono:

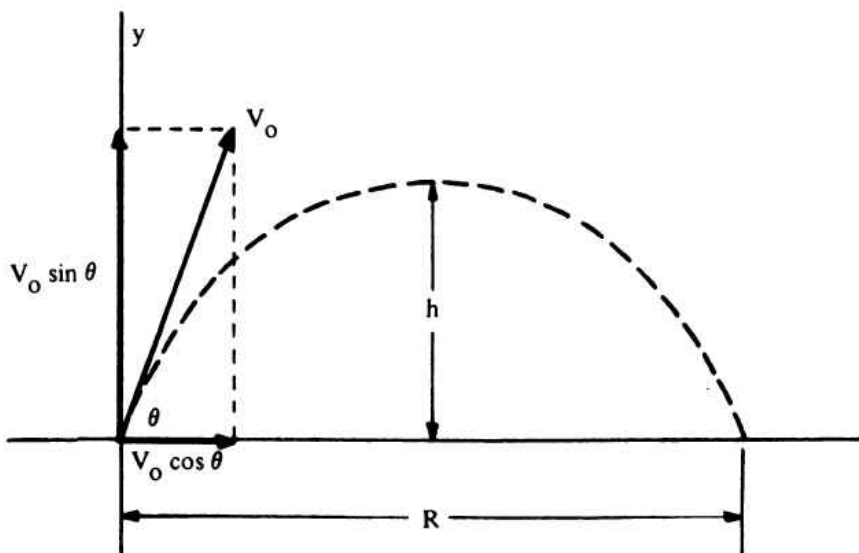
$$V_{ox} = V_o \cos\theta \ , \ V_{oy} = V_o \sin\theta$$

Siccome trascuriamo la resistenza dell'aria, la componente orizzontale della velocità rimane costante durante il moto. In qualche momento, essa è perciò:

$$V_x = V_{ox} = V_o \cos\theta = \text{costante} \quad (1)$$

La componente verticale del moto è costantemente portata verso il basso dall'accelerazione di gravità. Lo stesso avviene per un corpo sparato verso l'alto con una velocità iniziale V_0 sen θ. Ad un tempo "t" dall'inizio, la velocità verticale è:

$$V_y = V_{oy} - gt = V_o \sin\theta - gt \quad (2)$$

dove "g" è l'accelerazione di gravità.

Lo spazio percorso orizzontalmente è dato da:

$$x = V_{ox}t = (V_0 \cos\theta)\, t \qquad (3)$$

mentre lo spazio verticale è dato da:

$$\begin{aligned} y &= V_{oy}t - 1/2\, gt^2 \\ &= (V_0 \sin\theta)\, t - 1/2\, gt^2 \end{aligned} \qquad (4)$$

Il tempo necessario perché il proiettile ritorni alla stessa altezza iniziale si ricava dall'equazione (4) ponendo y=0. Da cui:

$$t = \frac{2\, V_0 \sin\theta}{g} \qquad (5)$$

Lo spazio orizzontale quando il proiettile ritorna alla sua altezza iniziale è detto portata orizzontale ("R"). Introducendo nell'equazione (3) il tempo necessario per raggiungere quel punto, otteniamo:

$$R = \frac{2\, V_0{}^2 \sin\theta \cos\theta}{g} \qquad (6)$$

Essendo 2sen θ cos θ =sen2 , l'equazione 6 diviene:

$$R = \frac{V_0{}^2 \sin 2\theta}{g} \qquad (7)$$

La portata orizzontale è allora proporzionale al quadrato della velocità iniziale per un dato angolo di elevazione. Poiché il valore massimo di sen2 .è 1, la massima portata orizzontale, Rmax, è V_0^2/g. Ma se sen2θ=1, allora 2θ=90° e θ=45°. Quindi la massima portata orizzontale, in assenza della resistenza dell'aria, si ottiene con un angolo di elevazione di 45°.

Dal punto di vista balistico, ciò che solitamente si vuole sapere è quale angolo di elevazione deve essere dato con una data velocità iniziale V_0 per colpire un obiettivo la cui posizione è nota. Supponendo che il cannone e l'obiettivo siano alla stessa altezza e che quest'ultimo sia ad una distanza R, l'equazione (7) può essere risolta per θ.

$$\begin{aligned} \theta &= 1/2 \sin^{-1}\left(\frac{Rg}{V_0{}^2}\right) \\ &= 1/2 \sin^{-1}\left(\frac{R}{R_{max}}\right) \end{aligned} \qquad (8)$$

Purché R sia inferiore alla gittata massima, questa equazione ha due soluzioni per valori di θ compresi tra 0° e 90°. Entrambi gli angoli forniscono la stessa gittata.

Naturalmente, la durata del volo e la massima altezza raggiunta sono entrambe maggiori per la traiettoria dell'angolo maggiore.
Supponiamo per esempio che la gittata massima della nostra arma sia 10.000 yarde e che l'obiettivo sia a 5.900 yarde:

$$\theta = 1/2 \sin^{-1} \frac{5,900}{10,000}$$

$$= 1/2\ 36^\circ$$

$$= 18^\circ, \text{ or } 90^\circ - 18^\circ = 72^\circ$$

Provate il gioco Gunner (Cannoniere) descritto più sotto. Utilizzate il metodo per tentativi ed errori per individuare e distruggere l'obiettivo (vedi anche l'esempio di esecuzione). Avete cinque possibilità per ogni obiettivo. Quanti colpi avete impiegato per distruggere tutti i cinque obiettivi? Vi è mai accaduto di non riuscire a distruggere un obiettivo con cinque tentativi?
Dalla discussione precedente dovrebbe esservi chiaro che la massima gittata si ottiene con 45° di elevazione, mentre i valori al di sotto o al di sopra di 45° forniscono una gittata minore.
La gittata massima del fucile varia tra 20.000 e 60.000 yarde ed il raggio di scoppio di una granata è di 100 yarde.
Potete anche determinare la corretta angolazione di tiro da tavole trigonometriche, con un regolo, una calcolatrice scientifica o un computer. Dovreste essere in grado di distruggere tutti gli obiettivi con un solo colpo. Che accade quando l'obiettivo è molto vicino? Potete usare sempre angoli interi?
Scrivete ora un programma che richieda la gittata massima del fucile e la distanza dell'obiettivo e poi calcoli la corretta angolazione di tiro. Dovrete risolvere due problemi per scrivere questo programma:

1. Il linguaggio Basic non dispone della funzione ARCSIN. La seguente formula può essere però d'aiuto.

$$\sin^{-1} x = \tan^{-1} \left(\frac{x}{\sqrt{1-x2}} \right)$$

2. Dovete effettuare una conversione da radianti a gradi sessagesimali.

```
5 printchr$(14)
8 printchr$(147)
10 print"Siete al comando di un plotone. Dovete  colpire entro 100 Yards"
50 print
70 r=int(40000*rnd(1)+20000)
80 print"gittata  ";r;"yards"
90 z=0
100 s1=0
110 t=int(r*(.1+.8*rnd(1)))
120 s=0
130 goto310
140 print"Min = 1 grado"
150 goto320
160 print"Max = 89 gradi"
170 goto320
180 print"Lungo di";abs(e);"yards"
190 goto320
200 print"Corto di";abs(e);"yards"
210 goto320
220 print"Obiettivo distrutto"
230 prints"Tentativi fatti"
240 print
250 s1=s1+s
260 ifz=4then450
270 z=z+1
280 print
290 print"Maggiore attivita' del nemico"
300 goto110
310 print"Obiettivo a";t;"yards"
320 rem
330 print:input"Elevazione";b
340 ifb>89then160
350 ifb<1then140
360 s=s+1
370 ifs<6then410
380 printchr$(147)
390 printtab(12)"Boom!":printtab(5)"Boom!":printtab(14)"Boom!":print
400 print"Il nemico ti ha colpito":goto480
410 b2=.035*b:i=r*sin(b2):x=t-i:e=int(x)
420 ifabs(e)<100then220
430 ife>100then200
440 goto180
450 printchr$(147);"Totale colpi=";s1
460 ifs1>18then480
470 print"Ottima mira!";:goto490
480 print"Torna ad addestrarti a Fort Sill!";
490 print:input"Ancora (y,n)";z$
500 ifz$="y"orz$="Y"then50
510 print:print"OK. Ritorna alla base"
520 stop
530 end
```

```
Siete al comando di un plotone. Dovete  colpire entro 100 Yards

gittata   35628 yards
Obiettivo a 22592 yards

Elevazione ? 10
Corto di 10375 yards

Elevazione ? 31
Lungo di 8914 yards

Elevazione ? 46
Lungo di 13009 yards

Elevazione ? 8
Corto di 12746 yards

Elevazione ? 5
Corto di 16388 yards

Elevazione ? 42

            Boom!
     Boom!
              Boom!

Il nemico ti ha colpito
Torna ad addestrarti a Fort Sill!
Ancora (y,n) ? y

gittata   56473 yards
Obiettivo a 32960 yards

Elevazione ? 2
Corto di 29010 yards

Elevazione ? 38
Lungo di 21884 yards

Elevazione ? 3
Corto di 27041 yards

Elevazione ? 45
Lungo di 23513 yards

Elevazione ? 26
Lungo di 11626 yards

Elevazione ? 42

            Boom!
     Boom!
              Boom!

Il nemico ti ha colpito
Torna ad addestrarti a Fort Sill!Ancora (y,n) ? n

OK. Ritorna alla base
```

9
Altri problemi

Ecco ora cinque problemi che non sembravano classificabili in alcuno dei capitoli precedenti. Il primo e l'ultimo sono giochi, benché anche la simulazione dell'atterraggio sulla luna possa essere inteso come un gioco. Uno è una simulazione dello smog, mentre l'ultimo programma calcola il deprezzamento secondo tre diversi metodi.

Il gioco dell'indovinare i numeri

Ecco un programma che gioca al popolare gioco dell'indovinare i numeri. Il computer sceglie un numero segreto compreso tra 1 e 100. A voi tocca cercare di indovinare il numero nel minor numero di tentativi.
Ci sono molti modi per cercare di indovinare il numero segreto. Fate giocare alcuni vostri amici ed osservate quale approccio utilizzano per trovare il numero segreto. Un approccio consiste nel partire dicendo 10. Se si rivela troppo basso, aumentare di 10 ad ogni tentativo finché il computer non vi dica che la vostra risposta è troppo alta. A questo punto partite dal numero precedente detto ed aumentate di 1 ad ogni tentativo. Questo è il metodo utilizzato per risolvere alcuni problemi nel capitolo sulla convergenza.
Un altro approccio consiste nel cercare di circoscrivere il numero entro un limite inferiore ed uno superiore e ridurre progressivamente i limiti finché il numero alla fine viene individuato. Due programmi sulla convergenza facevano uso di questo approccio.
Il metodo migliore è uno di questi due? Bene, questi metodi non sono male, soprattutto in confronto al partire da 1 e contare semplicemente verso il 100 finche non si giunge alla soluzione. C'è tuttavia un modo migliore, noto con il termine di ricerca binaria. In base a questa tecnica l'ambito di ricerca — in questo caso da 1 a 100 — viene diviso a metà, poi ancora a metà, e così via finché il numero segreto non viene scoperto. Fate questo gioco molte volte, utilizzando i diversi approcci. Alla lunga vedrete che la ricerca binaria è l'approccio più efficiente.
Alla linea 230 il programma vi dice che non dovreste avere bisogno di 7 tentativi. Perché? Se il numero segreto fosse tra 1 e 128, qual è il numero massimo di tentativi necessari per trovarlo? E se fosse tra 1 e 130, quale sarebbe il numero massimo di tentativi?
Modificare il programma in modo che scelga un numero segreto tra 1 e 10.000. Ora il limite superiore è 100 volte più alto rispetto al gioco iniziale che richiedeva un massimo di 7 tentativi per scoprire il numero segreto: quanti tentativi ci vorranno ora?
Provate a scrivere un programma in cui i ruoli del computer e del giocatore siano scambiati. In altre parole, il computer dovrà cercare di indovinare il vostro numero segreto compreso tra 1 ed un qualche limite superiore. Dopo ogni tentativo, dite al computer se il valore che vi ha dato è basso, alto o corretto.
Riuscite a scrivere il programma in modo tale che il computer possa accorgersi se state mentendo, cioè se gli state dando indicazioni incongruenti fra loro?

```
5 printchr$(14)
10 print"Ho un numero segreto tra 1 e 100.       Indovinalo"
20 print"Ti daro' qualche traccia"
30 print
40 print
50 c=0
60 n=int(rnd(1)*100)+1
70 c=c+1
80 printtab(10);"Prova";
90 inputg
100 ifn=gthen160
110 ifg>nthen140
120 print"Troppo basso"
130 goto70
140 print"Troppo alto"
150 goto70
160 print:print"Esatto in";c;"prove"
170 ifc>3then200
180 print"Sei stato fortunato"
190 goto240
200 ifc>7then230
210 print"Buon lavoro"
220 goto240
230 print"Dovresti farcela con 7 prove"
240 print
250 print
260 input"Un'altra partita (s/n)";a$
270 ifa$="s"ora$="S"then30
280 stop
```

```
Ho un numero segreto tra 1 e 100.       Indovinalo
Ti daro' qualche traccia


          Prova ? 50
Troppo basso
          Prova ? 65

Esatto in 2 prove
Sei stato fortunato


Un'altra partita (s/n) ? n


Ho un numero segreto tra 1 e 100.       Indovinalo
Ti daro' qualche traccia


          Prova ? 51
Troppo basso
          Prova ? 74
Troppo alto
          Prova ? 62
Troppo basso
          Prova ? 11
Troppo basso
          Prova ? 41

Esatto in 5 prove
Buon lavoro


Un altra partita (s/n) ? n
```

Tre metodi per calcolare il deprezzamento

Questo programma mostra come il capitale associato ad un macchinario si deprezzi, in base a tre metodi comunemente usati per calcolare il deprezzamento: linea retta, somma delle cifre dell'anno e diminuzione per raddoppio.
Il programma chiede il costo originale del prodotto, la sua durata prevista in anni (cioè il periodo di tempo sul quale va calcolato il deprezzamento), ed il valore previsto come rottame (o come usato) al termine di tale periodo.
Viene poi mostrata una tabella che illustra il deprezzamento annuale in base ad ognuno dei tre metodi.

```
5 printchr$(14)
10 print"Deprezzamento : 3 metodi"
20 print
30 input"Costo originale";c
50 input"Durata (anni)";l
70 input"Valore usato";s
90 print
100 v=c-s
110 d1=v/l
120 print"Il deprezzamento in lineare e' $";d1;"per anno":print
130 y=((l+1)/2)*l
140 z=l
150 print"            Somma     Diminuzione"
160 print"Anno        Cifre     per raddoppio"
165 print"----        -----     -------------"
170 forx=1tol
180 d2=v*(z/y)
190 d2=int(d2*100+.5)/100
200 z=z-1
210 d3=2*c/l
220 d3=int(d3*100+.5)/100
230 c=c-d3
240 printx;tab(5);d2;tab(17);d3
250 nextx
```

```
Deprezzamento : 3 metodi

Costo originale ? 8000
Durata (anni) ? 8
Valore usato ? 500

Il deprezzamento in lineare e' $ 937.5 per anno

          Somma       Diminuzione
Anno      Cifre       per raddoppio
----      -----       -------------
 1        1666.67       2000
 2        1458.33       1500
 3        1250          1125
 4        1041.67       843.75
 5        833.33        632.81
 6        625           474.61
 7        416.67        355.96
 8        208.33        266.97
```

La simulazione dello smog

Questo programma è un adattamento del modello sullo smog originariamente scritto da Herbert Peckham. Il modello ipotizza che il traffico automobilistico sia l'unica fonte di smog, il che è un'ipotesi piuttosto limitata. Ipotizza anche che la maggior parte del traffico avvenga durante le ore diurne e che il volume di traffico sia molto basso (in realtà, zero) di notte. Lo smog generato dalle automobili viene dissipato dalle condizioni atmosferiche, le quali variano a seconda della luce del sole, la temperatura ed il tempo. L'utente può specificare tutte queste condizioni.
Il modello potrebbe essere migliorato significativamente prendendo in considerazione altre fonti di smog, variando il traffico a seconda dell'ora del giorno ed infine consentendo variazioni giornaliere dei fattori meteorologici.
Ciononostante, il programma è interessante ed istruttivo anche in questa forma rudimentale.
L'istruzione 400 genera una curva che riproduce il livello di smog. In certe condizioni il livello di smog raggiunge valori che non possono essere tracciati nella curva perché maggiori dell'ampiezza dello schermo (e della stampante).
In questo caso può essere opportuno cancellare la routine che traccia la curva.
Una soluzione più elegante sarebbe invece di stimare il valore massimo del livello di smog in base ai fattori di input e calcolare così un appropriato fattore di moltiplicazione in fase di tracciamento della curva.

```
5 printchr$(14)
10 printchr$(147);"Simulazione di smog"
20 print
30 t=7
35 print
40 print"Auto per miglio ":print"Los Angeles   200":print"Detroit        100"
45 print"Tulsa           25"
60 print"Smog           ";:inputc
65 print
70 print"Fattore di generazione":print"1950 auto   .005":print"1975 auto   .001"
75 print"Bus            .010"
80 print"Smog          ";:inputl1
90 print
100 print"Smog di giorno    ":print"in percento per ora"
105 print"sereno       .02":print"Nuvoloso    .01"
110 print"Smog         ";:inputr1
120 print
130 print"Smog di notte    ":print"caldo          .05":print"Freddo       .10"
140 print"Smog         ";:inputr2
150 print
160 print"Dispersione (% per ora)":print"Vento e pioggia  1.00"
165 print"Calmo e secco    0.01"
170 print"Smog                ";:inputr3
180 print
190 input"Tabella o Plot (t/p)";a$
200 print"Ora   livello smog"
210 r=r1
220 k1=11
230 t1=int((t-6)/12)
240 ift1/2=int(t1/2)then270
```

```
250 k1=0
260 r=r2
270 s=s+k1*c*10-r*s-r3*s
280 ifs<=0thens=0
290 t=t+1
300 t3=int(t/12)
310 t2=t-t3*12+1
320 x=int(1000*s+.5)/1000
330 ift3/2=int(t3/2)then350
340 printt2;"PM";tab(7);:goto360
350 printt2;"AM";tab(7);
360 ifa$="p"ora$="P"then390
370 printx
380 goto210
390 ifx>17thenx=17
400 printtab(6+x);"*"
410 goto210
```

```
Simulazione di smog


Auto per miglio
Los Angeles    200
Detroit        100
Tulsa          25
Smog         ? 100

Fattore di generazione
1950 auto    .005
1975 auto    .001
Bus          .010
Smog       ?   .001

Smog di giorno
in percento per ora
sereno       .02
Nuvoloso     .01
Smog       ? .01

Smog di notte
caldo        .05
Freddo       .10
Smog       ? .1 0
```

```
Dispersione (% per ora)
Vento e pioggia  1.00
Calmo e secco    0.01
Smog         ? .05

Tabella o Plot (t/p) ? t
Ora   livello smog
 9 AM       1
 10 AM       1.94
 11 AM       2.824
 12 AM       3.654
 1 PM       4.435
 2 PM       5.169
 3 PM       5.859
 4 PM       6.507
 5 PM       7.117
 6 PM       7.69
 7 PM       8.228
 8 PM       7.817
 9 PM       7.426
 10 PM       7.055
 11 PM       6.702
 12 PM       6.367
 1 AM       6.049
 2 AM       5.746
 3 AM       5.459
 4 AM       5.186
 5 AM       4.927
 6 AM       4.68
 7 AM       4.446
 8 AM       5.18
 9 AM       5.869
 10 AM       6.51700001
 11 AM       7.126
```

```
Simulazione di smog


Auto per miglio
Los Angeles   200
Detroit       100
Tulsa          25
Smog        ? 100

Fattore di generazione
1950 auto    .005
1975 auto    .001
[illegible]          .010
Smog       ?  .001

Smog di giorno
in percento per ora .
sereno       .02
Nuvoloso     .01
Smog       ? .01

Smog di notte
caldo        .05
Freddo       .10
Smog .     ? .1 0
```

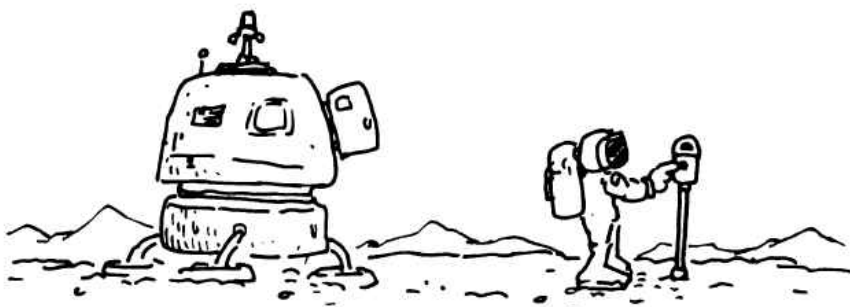
```
Dispersione (% per ora)
Vento e pioggia  1.00
Calmo e secco    0.01
Smog        ? .05

Tabella o Plot (t/p) ? p
Ora   livello smog
 9 AM              •
 10 AM              •
 11 AM               •
 12 AM                •
 1 PM                 •
 2 PM                  •
 3 PM                  •
 4 PM                   •
 5 PM                    •
 6 PM                    •
 7 PM                     •
 8 PM                    •
 9 PM                    •
 10 PM                    •
 11 PM                   •
 12 PM                   •
1 AM                    •
2 AM                   •
3 AM                   •
4 AM                   •
5 AM                  •
6 AM                  •
7 AM                  •
8 AM                   •
9 AM                   •
10 AM                    •
11 AM                     •
```

Simulazione dell'allunaggio

Questo programma è una delle più popolari simulazioni per computer, oggi disponibile in molte versioni: questa è un adattamento del programma originale scritto nel 1969.
Il programma rappresenta un'esatta simulazione di un modulo di allunaggio Apollo durante la discesa finale. Questa parte della discesa veniva normalmente controllata dal computer di bordo, eventualmente sostituito da un altro computer sulla navicella orbitante ed un altro ancora sulla Terra. Tuttavia, per esercitare le vostre conoscenze di fisica e per rendere interessante il gioco, ipotizziamo che tutti e tre i computer abbiano avuto un malfunzionamento simultaneo. È perciò soltanto nelle vostre mani la riuscita dell'atterraggio del modulo spaziale.
Per effettuare un atterraggio morbido, potete modificare la quantità di carburante bruciato dai retrorazzi ogni dieci secondi. Potete tenerli spenti (tasso di consumo = 0) o farli bruciare ad un tasso compreso tra 9 e 200 libbre di carburante al secondo. L'alimentazione del motore avviene ad 8 libbre alla volta, quindi non sono possibili valori da 1 a 7 libbre. Avete a disposizione 16.500 libbre di carburante. Questa quantità è superiore di 500 libbre a quella effettivamente disponibile su un vero LEM, il che vi lascerà un piccolo margine di errore. Quando sarete diventati più abili, modificate l'istruzione 130 in N=16000 così da avere una simulazione più aderente alla realtà. Il peso della capsula è di 33.000 libbre.
Non che questo sia il modo per arrivare, ma se non accendeste del tutto i razzi, il tempo stimato necessario per una discesa in caduta libera è di 120 secondi per l'impatto (con un gran botto).
Buona fortuna!

```
5 printchr$(14)
10 printchr$(147);"      Allunaggio"
15 print"      di David Ahl"
20 print:print"Malfunzionamento del computer - Prendi  il controllo manuale"
30 print
40 print"Peso capsula -- 32500 lbs"
45 print"Peso carb.   -- 16500 lbs"
50 print
60 print"Accendi i retrorazzi ogni 10 secondi o"
70 print"a un valore tra 8 & 200 lbs per sec."
80 print
90 print"Miglia   MPH   Carb.   Tempo   Retro"
95 print"------   ---   -----   -----   -----"
100 a=120
110 v=1
120 m=33000
130 n=16500
140 g=.001
150 z=1.8
160 p=int(3600*v+.5)
170 ifabs(p)>=1then190
180 p=int((3600*v+.5)*1000)/1000
190 ifa<10then210
```

```
200 d=int(a+.5):goto240
210 ifa<1then230
220 d=(int(10*a+.5))/10:goto240
230 d=(int(1000*a+.5))/1000
240 printd;tab(8);p;tab(14);int(m-n);tab(22);l;tab(30);:inputk
260 ifk>200then290
270 ifk>7then300
280 ifk=0then300
290 print"Impossibile. Riprova":goto240
300 t=10
320 ifm-n<.001then430
330 ift<.001then160
340 s=t
350 ifm>n+s*kthen370
360 s=(m-n)/k
370 gosub770
380 ifi<=0then630
390 ifv<=0then410
400 ifj<0then690
410 gosub570
420 goto320
430 print"Carburante esaurito a";l;"secondi"
440 s=(-v+sqr(v*v+2*a*g))/g
450 v=v+g*s
460 l=l+s
470 w=3600*v
480 print"sulla luna a";int(l);"secondi"
482 print"Velocita' di impatto";
485 printint(100*w+.5)/100;"MPH"
490 ifw>1.2then510
500 print"Atterraggio perfetto":end
510 ifw>10then530
520 print"Oooooomph!  Che botta!   Hai bisogno di piu' pratica !":end
530 ifw>60then550
540 print"Severi danni alla navetta.Devi attendere gli aiuti":end
550 print"Spiacente ! Nessun sopravvissuto. Tutto distrutto !":end
560 print"Hai creato un nuovo cratere profondo ben";int(w*278);"piedi  !":end
570 l=l+s
580 t=t-s
590 m=m-s*k
600 a=i
610 v=j
620 return
630 ifs<.005then470
640 d=v+sqr(v*v+2*a*(g-z*k/m))
650 s=2*a/d
660 gosub770
670 gosub570
680 goto630
690 w=(1-m*g/(z*k))/2
700 s=m*v/(z*k*(w+sqr(w*w+v/z)))+.05
710 gosub770
720 ifi<=0then630
730 gosub570
740 ifj>0then320
750 ifv>0then690
760 goto320
770 q=s*k/m
780 j=v+g*s-z*q*(1+q*(.5+q*(1/3+q*(.25+q/5))))
790 i=a-g*s*s/2-v*s+z*s*q*(.5+q*(1/6+q*(1/12+q/20)))
800 return
```

```
        Allunaggio
        di David Ahl

Malfunzionamento del computer - Prendi il controllo manuale

Peso capsula -- 32500 lbs
Peso carb.   -- 16500 lbs

Accendi retrorazzi ogni 10 secondi o
a un valore tra 8 e 200 lbs per sec.

Miglia   MPH    Carb.    Tempo    Retro
------   ---    -----    -----    -----
  120    3600   16500     0       ? 0
  110    3636   16500     10      ? 0
  100    3672   16500     20      ? 0
   90    3708   16500     30      ? 50
   79    3645   16000     40      ? 50
   69    3581   15500     50      ? 50
   59    3515   15000     60      ? 100
   50    3341   14000     70      ? 100
   41    3161   13000     80      ? 100

   32    2974   12000     90      ? 100
   24    2779   11000    100      ? 200
   17    2325    9000    110      ? 200
   11    1832    7000    120      ? 200
  7.1    1292    5000    130      ? 200
  4.4     695    3000    140      ? 200
  3.3      30    1000    150      ? 150
Carburante esaurito a 156.666667 secondi.
sulla luna a 372 secondi
Velocita' di impatto 448.42 MPH
Spiacente ! Nessun sopravvissuto. Tutto distrutto !
```

Hammurabi

Hammurabi è uno tra i più apprezzati giochi per computer, di tutti i tempi. Da una parte, può essere considerato un gioco, ma dall'altra parte si tratta di un affascinante simulazione di baratto e di conduzione direttiva.
Con Hammurabi cercate di gestire l'antica città-stato della Sumeria. L'economia della città-stato ruota intorno ad una sola cosa: il raccolto annuale di grano (probabilmente soia).
Ogni anno, dovete decidere quanti bushels (*) di grano volete fornire per l'alimentazione del vostro popolo (scoprirete presto quanto necessita a una persona per sopravvivere), quanti volete usarne come semi per piantare il raccolto per l'anno successivo, quanti volete usarne per acquistare altra terra dalle città-stato a voi vicine e quanti volete accumularne in magazzino.
Naturalmente, se vi capita un cattivo raccolto o se i ratti invadono le ceste in cui avete immagazzinato il vostro grano, dovrete vendere parte della vostra terra per avere abbastanza grano per evitare che il vostro popolo muoia di fame, o per seminare la terra l'anno successivo. Sfortunatamente, i disastri sembrano sempre pronti a colpire, costringervi a vendere la terra quando il prezzo è basso; ma ciò non è molto diverso da quanto avviene nel mondo reale.
Molti incominciano a giocare con nobili ambizioni. Tuttavia, poco più tardi, incominciano a desiderare che una pestilenza decimi la loro crescente popolazione. Oppure fanno deliberatamente morire di fame una parte del popolo per mantenere le cose in equilibrio.
Durante gli anni, questo gioco, più di ogni altro, ha generato una quantità di imitazioni, estensioni e modifiche. In verità, molti produttori hanno preso il mio originale senza nemmeno apportare alcuna modifica, l'hanno messo in una bella scatola e gli hanno posto anche un bel prezzo. Non accettate imitazioni! Eccovi il gioco originale (con il dialogo leggermente abbreviato) perché possiate usarlo sul vostro computer.
Se volete sperimentare delle modifiche, ecco alcuni suggerimenti. Nel gioco originario le pestilenze accadono casualmente per il 15% del tempo; abbassate questa percentuale al 10% o al 5%. La popolazione richiede ora una quantità fissa di cibo; variate leggermente questa quantità, anno dopo anno. Consentite la costruzione di silos a prova di ratto, ma che devono costare una grossa cifra. Introducete un'industria mineraria, oltre all'attività agricola. E che ne dite della pesca e del turismo? Fate correre liberatamente la vostra immaginazione. Sperimentate! Divertitevi!

(*): il bushel è una misura di capacità usata per i cereali e la frutta, pari a 8 galloni, cioè a litri 32,50 circa (N.d.T.).

```
5 printchr$(14)
10 printchr$(147);"Prova a governare l'antica Sumeria  per 10 anni."
30 d1=0:p1=0
40 z=0:p=95:s=2800:h=3000:e=h-s
50 y=3:a=h/y:i=5:q=1
60 d=0
70 print:z=z+1
80 print" Hammurabi : anno",z
90 printd;"morti   ";i;"nati"
100 p=p+i
110 ifq>0then140
120 p=int(p/2)
130 print" Orribile pestilenza"
140 print" Popolazione";p
150 print" Possiedi";a;"acri":print" che producono";y;"Brushels per acro."
180 print" I topi divorano";e;"Brushels"
190 prints;"Brushels immagazzinati"
200 ifz=11then840
210 c=int(10*rnd(1)):y=c+17
220 print" Terra=";y;"Brushels/acro"
230 input" Acri comprati";q
250 ifq<0then810
260 ify*q<=sthen290
270 gosub730
280 goto230
290 ifq=0then320
300 a=a+q:s=s-y*q:c=0
310 goto380
320 input" Acri venduti";q
330 ifq<0then810
340 ifq<athen370
350 gosub760
360 goto320
370 a=a-q:s=s+y*q:c=0
380 print
390 input" Brushels per cibo";q
400 ifq<0then810
410 ifq<=sthen440
420 gosub730
430 goto390
440 s=s-q:c=1
450 print
460 input" Acri per semina";d:ifd=0then570
470 ifd<0then810
480 ifd<=athen500
490 goto460
500 ifint(d/2)<=sthen530
510 gosub730
520 goto460
530 ifd<10*pthen560
540 print" Non c'e' abbastanza gente"
550 goto460
560 s=s-int(d/2)
570 gosub790
580 y=c:h=d*y:e=0
590 gosub790
600 ifint(c/2)<>c/2then620
620 s=s-e+h
630 gosub790
640 i=int(c*(20*a+s)/p/100+1)
650 c=int(q/20)
660 q=int(10*(2*rnd(1)-.3))
670 ifp<cthen60
680 d=p-c:ifd>.45*pthen710
```

```
690 p1=((z-1)*p1+d*100/p)/z
700 p=c:d1=d1+d:goto70
710 print:printd;"persone sono morte di fame !"
720 print" Sei destituito !":goto1030
730 print" Ripensaci, hai solo";s;"Brushels"
750 return
760 print" Ripensaci, hai solo";a;"acri"
780 return
790 c=int(rnd(1)*5)+1
800 return
810 print" Hammurabi non ce la faccio. Trovati un  altro aiutante !":goto1030
840 print" In 10 anni";p1;"%"
850 print" Morti di fame. Totale";d1;"morti !"
870 l=a/p
880 print" Hai incominciato con 10 acri/persona e  hai finito con";l;"acri/perso
910 print
920 ifp1>33then720
930 ifl<7then720
940 ifp1>10then990
950 ifl<9then990
960 ifp1>3then1010
970 ifl<10then1010
980 print" Sei stato fantastico !":goto1030
990 print" Hai la mano pesante.":print" Il popolo si ribella !":goto1030
1010 print" Non male, ma potresti andar meglio"
1030 print:print" Addio....per ora"
1050 end
```

```
Prova a governare l'antica Sumeria  per 10 anni.

 Hammurabi : anno 1
 0 morti    5 nati
 Popolazione 100
 Possiedi 1000 acri
 che producono 3 Brushels per acro.
 I topi divorano 200 Brushels
 2800 Brushels immagazzinati
 Terra= 22 Brushels/acro
 Acri comprati ?  0
 Acri venduti ?  0

 Brushels per cibo ?  1500

 Acri per semina ?  500

 Hammurabi : anno 2
 25 morti    10 nati
 Popolazione 85
 Possiedi 1000 acri
 che producono 3 Brushels per acro.
 I topi divorano 0 Brushels
 2550 Brushels immagazzinati
 Terra= 23 Brushels/acro
 Acri comprati ?  0
 Acri venduti ?  0

 Brushels per cibo ?  1500

 Acri per semina ?  500
```

```
Hammurabi : anno 3
10 morti    3 nati
Popolazione 78
Possiedi 1000 acri
che producono 5 Brushels per acro.
I topi divorano 0 Brushels
3300 Brushels immagazzinati
Terra= 26 Brushels/acro
Acri comprati ?  0
Acri venduti ?  200

Brushels per cibo ?  1500

Acri per semina ?  500

Hammurabi : anno 4
3 morti    13 nati
Orribile pestilenza
Popolazione 44
Possiedi 800 acri
che producono 4 Brushels per acro.
I topi divorano 0 Brushels
8750 Brushels immagazzinati
Terra= 26 Brushels/acro
Acri comprati ?  0
Acri venduti ?  0

Brushels per cibo ?  2000


Acri per semina ?  800
Non c'e' abbastanza gente
Acri per semina ?  500
Non c'e' abbastanza gente
Acri per semina ?  300

Hammurabi : anno 5
0 morti    6 nati
Orribile pestilenza
Popolazione 25
Possiedi 800 acri
che producono 3 Brushels per acro.
I topi divorano 0 Brushels
7500 Brushels immagazzinati
Terra= 17 Brushels/acro
Acri comprati ?  0
Acri venduti ?  20

Brushels per cibo ?  2000

Acri per semina ?  500
Non c'e' abbastanza gente
Acri per semina ?  0

Hammurabi : anno 6
0 morti    9 nati
Popolazione 34
Possiedi 780 acri
che producono 3 Brushels per acro.
I topi divorano 0 Brushels
5840 Brushels immagazzinati
Terra= 24 Brushels/acro
Acri comprati ?  80
```

```
Brushels per cibo ?  2500 ,

Acri per semina ?  500
Non c'e' abbastanza gente
Acri per semina ?  0

Hammurabi : anno 7
0 morti    11 nati
Popolazione 45
Possiedi 860 acri
che producono 3 Brushels per acro.

I topi divorano 0 Brushels
1420 Brushels immagazzinati
Terra= 23 Brushels/acro
Acri comprati ?  0
Acri venduti ?  2000
Ripensaci, hai solo 860 acri
Acri venduti ?  800

Brushels per cibo ?  30

Acri per semina ?  0

44 persone sono morte di fame !
Sei destituito !

Addio....per ora
```

Bibliografia

La rivista a cui si fa riferimento nel testo è:
— *Creative Computing*. si tratta della rivista numero uno nel campo del software e delle applicazioni per tutti i piccoli computer. Contiene articoli, tutorials, applicazioni ed ampie valutazioni.
Tra i libri a cui si fa riferimento nel testo:
— *Computers in mathematics*: A Sourcebook of Ideas. Centinaia di idee, verificate in classi scolastiche, sull'uso del computer per imparare la matematica.
— *Computers in Science and Social Studies*. Dozzine di programmi di simulazione in biologia, ecologia, fisica e gestione dei sistemi del mondo reale.

Commodore 64
IDEA BOOK

Questo "Idea Book" contiene dozzine di modi per ottenere il meglio dal vostro computer per risolvere problemi di diverso tipo. I 50 programmi in esso contenuti sono pronti per essere eseguiti e dimostrano molteplici tecniche per risolvere problemi di matematica, scienza e affari.
Nei dieci capitoli del libro si affrontano la soluzione di problemi con formule e per tentativi successivi, i problemi di convergenza, la ricorsività, il calcolo, la probabilità, la geometria, le scienze, le simulazioni, ecc.
Alcuni problemi mostrano la potenza del computer; altri ne illustrano invece i limiti. È importante familiarizzarsi sia con i punti di forza che con quelli di debolezza degli strumenti, così da saper riconoscere per quali tipi di lavoro sono più adatti.

L'autore, David H. Ahl, ha incominciato ad interessarsi all'uso dei computer nel 1957. Ha scritto 16 libri ed è il fondatore ed il direttore editoriale delle riviste "Creative Computing", "SYNC" e "Video & Arcade Games".

.0017647

£.18.000

Editrice Il Rostro - **Commodore 64 IDEA BOOK** - David H. Ahl